FlameRobin: Ferramenta gratuita para administração de bancos de dados Firebird

Clube Delphi

Delphi Prism
Entenda as Expressões Lambda

Delphi Prism
Personalizando aplicações WPF

Ano 11:: Edição 132 :: R$ 11,90

DevMedia

Nota fiscal Eletrônica 2.0

Morfik
Desenvolvimento Web com Object Pascal

Bancos de Dados Gratuitos
Conheça as versões Express do Oracle, SQL Server e DB2

ISSN 1517990-7

Dificuldades em encontrar conteúdo sobre .Net para iniciantes?

A DevMedia simplifica pra você!

easy .net magazine

Easy .Net Magazine:

A primeira revista digital feita especialmente para quem está dando os primeiros passos em .Net!

Você encontrará exemplos e modelos fáceis e práticos, noções introdutórias e conceitos básicos.

Conheça as vantagens:

- Vídeos e apresentações "embutidos" nas páginas;
- Ilustrações, diagramas e modelos apresentados em PowerPoint;
- Entrevistas em vídeo;
- Códigos-fonte que podem ser copiados com CTRL+C/CTRL+V.

A Easy.Net não é feita em PDF ou Flash!
Ela utiliza um aplicativo 100% online.

DevMedia group

Acesse agora a edição Nº 1 totalmente grátis e comprove

www.devmedia.com.br/easynet

Easy.Net Magazine, seu curso inicial de .Net!

Sumário

Delphi

Boa Ideia — **06 – Conheça as novidades da Nota Fiscal Eletrônica 2.0**
[*Jederson Zuchi*]

Boa Ideia — **13 – Conheça o FlameRobin, ferramenta gratuita para administração de bancos de dados Firebird**
[*Fabrício Hissao Kawata*]

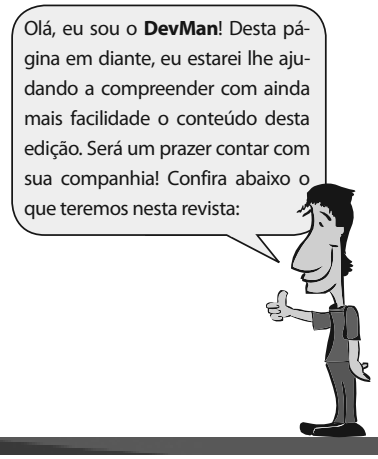
Delphi — **21 – Personalizando aplicações WPF no Delphi Prism**
[*Fabrício Hissao Kawata*]

Tutorial — **31 – Conheça o Morfik, IDE para desenvolvimento Web usando Object Pascal**
[*Carlos Eduardo Paulino*]

Mini-Curso — **39 – Bancos de Dados Gratuitos – Parte 2**
[*Vitor Luiz Rubio*]

Boas Práticas

Boas Práticas — **48 – Expressões Lambda**
[*Paulo Quicoli*]



Dê seu feedback sobre esta edição!

A ClubeDelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre esta edição, artigo por artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Conhecimento faz diferença!

Faça já sua assinatura digital ! | www.devmedia.com.br/es

Faça um *upgrade* em sua carreira

Em um mercado cada vez mais focado em qualidade, ter conhecimentos aprofundados sobre requisitos, metodologia, análises, testes, entre ouutros, pode ser a diferença entre conquistar ou não uma boa posição profissional. Sabendo disso a DevMedia lançou uma publicação totalmente especializada em Engenharia de Software. Todos os meses você pode encontrar artigos sobre Metodologias Ágeis; Metodologias tradicionais (document driven); ALM (application lifecycle); SOA (aplicações orientadas a serviçoes); Análise de sistemas; Modelagem; Métricas; Orientação à Objetos; UML; testes e muito mais. **Assine Já!**

DevMedia

Clube Delphi

Ano 11 - 132ª Edição 2011 - ISSN 1517990-7 Impresso no Brasil

Corpo Editorial

Editor Geral
Guinther Pauli
guinther@devmedia.com.br

Editor Técnico
Paulo Quicoli
pauloquicoli@devmedia.com.br

Leandro Daniel
contato@leandrodaniel.com

Equipe Editorial
Vitor Luiz Rubio
vitorrubio@gmail.com

Rafael Stavarengo
faelsta@gmail.com

Jornalista Responsável
Kaline Dolabella - JP24185

Atendimento ao Leitor
A DevMedia conta com um departamento exclusivo para o atendimento ao leitor. Se você tiver algum problema no recebimento do seu exemplar ou precisar de algum esclarecimento sobre assinaturas, exemplares anteriores, endereço de bancas de jornal, entre outros, entre em contato com:

www.devmedia.com.br/mancad
(21) 3382-5038

Publicidade
Para informações sobre veiculação de anúncio na revista ou no site e para fechar parcerias ou ações específicas de marketing com a DevMedia, entre em contato com:

publicidade@devmedia.com.br

Distribuição
FC Comercial e Distribuidora S.A
Rua Teodoro da Silva, 907
Grajaú - RJ - 206563-900

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique a vontade para entrar em contato com os editores e dar a sua sugestão!
Se você estiver interessado em publicar um artigo na revista ou no site ClubeDelphi, entre em contato com os editores, informando o título e mini-resumo do tema que você gostaria de publicar:

Guinther Pauli - Editor da Revista
guinther@devmedia.com.br

EDITORIAL

Nesta edição trazemos uma matéria especial para os desenvolvedores Delphi, sobre o que há de novo da Nota Fiscal Eletrônica 2.0, bem como o suporte do mais popular conjunto de componentes para Delphi, o AcbrNfe. A NFE é um projeto que foi instituído pelo governo federal há um bom tempo, facilitando a vida das empresas na redução de papéis armazenados e também uma maior interação entre sistemas. A nota fiscal eletrônica vem evoluindo gradativamente para tentar cobrir o maior número possível de operações, e também embutir as regras de negócios na própria validação da nota. No artigo do Jederson Zuchi, o leitor verá o que há de diferente em relação à sua versão anterior e como os desenvolvedores Delphi podem utilizar mecanismos já prontos – componentes – para agilizar uma solução para seus clientes.

O Fabrício Hissao Kawata traz duas matérias nesta edição. Conheça o FlameRobin, ferramenta open source voltada exclusivamente para a administração e manipulação de bancos de dados Firebird. Saiba como fazer procedimentos cotidianos como manipulação de tabelas, stored procedures, views, backup, restore, geração de script DDL e dados para testes. Em sua outra matéria, o autor aprofunda um importante tema sobre uma tecnologia que agora está presente na vida dos desenvolvedores Delphi, através do Prism. Estamos falando do WPF, Windows Presentation Foundation, tema que temos trazido com frequência aqui na revista. Nesta edição, o leitor verá como trabalhar com a personalização de aplicações, usando estilos e templates.

Ainda nesta edição, temos um artigo sobre uma interessante ferramenta que pode facilitar muito a vida dos desenvolvedores Delphi que necessitam criar soluções para Web. O Morfik, que usa uma linguagem baseada no Object Pascal, permite criar soluções Web de última geração, com interfaces ricas, exigindo o mínimo de esforço e codificação por parte dos desenvolvedores, mesmo aqueles menos experientes no desenvolvimento para a plataforma Web.

O Vitor Rubio continua a série de artigos que trata do uso de Bancos de Dados Gratuitos, mostrando aos programadores Delphi que existem outras excelentes soluções free além do já consagrado e consolidado Firebird. Mesmo os bancos mais robustos, como o Oracle e SQL Server, possuem versões Express que podem ser usadas livremente em aplicativos Delphi. Inclusive, para soluções Web, onde por exemplo o SQL Server é uma ferramenta mais indicada. Além destes bancos, temos ainda uma outra grande opção, o DB2 Express, da IBM, que assim como o Oracle, roda em várias plataformas.

Certamente o leitor já deve ter se deparado, seja no C# ou mesmo o Delphi Prism, com uma Expressão Lambda. Nós como desenvolvedores Delphi, em um primeiro contato, não temos como negar que a expressão é bastante estranha, principalmente para quem não está acostumado com o uso de delegates e métodos anônimos. E como o uso desse tipo de construção está se tornando um padrão na plataforma, nada melhor do que compreender como surgiram e quando utilizar.

Guinther Pauli
guinther@devmedia.com.br
Certificado Delphi 3,5,6,7,2005,2006,Web,Linux
Microsoft Certified: MCP, MCAD, MCSD.NET, MCTS, MCPD (ASP.NET,C#)
guintherpauli.blogspot.com - twitter.com/guintherpauli

ClubeDelphi PLUS

A revista ClubeDelphi é parte integrante da assinatura ClubeDelphi PLUS.
Para mais informações sobre o pacote PLUS, acesse:
http://www.devmedia.com.br/clubedelphi/portal.asp

DevMedia

Sistema de créditos DevMedia

Conteúdo **completo** ao seu alcance

Com o sistema de créditos DevMedia você pode comprar individualmente mais de 4.000 vídeo aulas, 8000 artigos, além de palestras e vídeos com formações completas.

TODAS as vídeo aulas que você sempre quis ver a partir de

R$ 0,99

ECONOMIA: você não precisa assinar 8 produtos diferentes para ter acesso ao conteúdo completo do site!

Saiba mais sobre o Sistema de Créditos!

www.devmedia.com.br/creditos

DevMedia

Nota Fiscal Eletrônica

A nota fiscal eletrônica é um projeto que foi instituído pelo governo federal, há um bom tempo, e vem dando certo, facilitando a vida das empresas no que tange a redução de papéis armazenados e também uma maior interação entre sistemas, como é o caso dos sistemas E.D.I. (Eletronic Data Interchange). Como qualquer outra tecnologia, a nota fiscal eletrônica vem evoluindo gradativamente para tentar cobrir o maior número possível de operações, e também, embutir as regras de negócios na própria validação da nota.

Quando surgiu a primeira versão da nota fiscal eletrônica, ela era basicamente um arquivo XML contendo as informações necessárias que o manual de orientação solicitava para cada operação. Não havia nenhuma regra de validação desses dados e a nota fiscal poderia ficar, por exemplo, com o total da nota menor do que o valor dos produtos, mesmo não tendo nenhum outro valor que se efetua o desconto. Concomitante a isto, a parte técnica foi evoluindo também, foram adicionadas mais opções para as validações, no arquivo XML incluídos alguns campos e então foi lançado o Schema 2.0 da NFe que é o padrão que está em vigor.

Neste schema 2.0 foi incluída a parte que diz respeito às empresas que têm regime de tributação no simples nacional, e para estas empresas, foram adicionadas novas tags para que as operações acobertadas por este regime de tributação pudessem acontecer. Dentre as tags incluídas o maior destaque foi para o código de CSOSN (código de situação da operação em simples nacional). Este conjunto de tags que será demonstrado mais à frente tem por objetivo destacar as operações que são feitas por empresas enquadradas no simples nacional, diferenciando-se do antigo schema 1.10 que havia uma nota técnica mostrando como fazer para empresas que estavam neste regime de tributação.

Conhecendo o Schema 2.0

É interessante abordar as alterações nesse schema em relação ao schema anterior para poder entender as validações e os possíveis campos que devem ser preenchidos quando for confeccionar a nota fiscal eletrônica na versão 2.0. Vale ressaltar que a melhor forma de fazer o preenchimento ou a implementação da nota fiscal eletrônica, é sempre utilizar a versão do manual do contribuinte que pode ser baixado acessando o site da nota fiscal eletrônica, onde o link se encontra na seção correspondente deste artigo.

A principal diferença deste registro em relação ao outro é justamente a inclusão do regime de tributação do simples nacional. Para isto nesta versão, foi colocado um campo na Tag destinada a identificação do emitente da NFe (<emit>), a tag incluída em questão é a tag <CRT>. Ela tem por objetivo informar no arquivo de nota fiscal eletrônica se o contribuinte que está emitindo é optante do simples nacional, simples nacional com excesso de renda bruta, e ou regime de tributação convencional. Esta informação é bastante importante para definir quais tags serão informadas na parte dos impostos, já que há uma diferença entre o preenchimento de um ou de outro no que tange à parte dos impostos. Esta tag de CRT é de preenchimento obrigatório e o não preenchimento deste pode gerar um erro de validação do XML.

Quando a tag de código do regime tributário estiver preenchida como tributação normal, as tags que eram preenchidas na versão do schema 1.10 continuam sendo preenchidas normalmente (tags icms00, icms10, icms20, etc). No entanto, se o regime de tributação da empresa for simples nacional, deve-se preencher outras tags dentro da tag icms. As tags que devem ser preenchidas são do ICMSSN, que são destinadas a empresas no simples nacional. A seguir pode-se verificar na **Figura 1** como deve ser o preenchimento do ICMS para o Simples Nacional. Já a **Figura 2** apresenta a tributação normal.

É importante ressaltar que as tags demonstradas são reais, no entanto podem conter variações baseadas em cada regra de tributação que a empresa pode ter.

Resumo DevMan

De que trata o artigo:

Este artigo trata do funcionamento da nota fiscal eletrônica, bem como demonstração do componente AcbrNfe.

Em que situação o tema é útil:

Este tema é bastante útil para quem quer conhecer a nota fiscal eletrônica além de quem precisa desenvolver as rotinas e não sabe por onde começar. São abordadas as melhores praticas utilizando o componente ACBrNFe.

Nota Fiscal Eletrônica 2.0:

O governo federal tem apostado em uma iniciativa que veio para ficar, a nota fiscal eletrônica. Assim como todo e qualquer projeto, ela passa por mudanças e avanços. Neste artigo é mostrado o que há de diferente em relação à sua versão anterior e como os desenvolvedores Delphi podem utilizar mecanismos já prontos – componentes – para agilizar uma solução para seus clientes. Através de exemplos, que estão disponíveis para download, será possível abordar o trâmite envolvido.

2.0

Jederson Zuchi

```
- <imposto>
   - <ICMS>
      - <ICMSSN102>
           <orig>0</orig>
           <CSOSN>102</CSOSN>
        </ICMSSN102>
     </ICMS>
```

Figura 1. Simples Nacional

```
- <imposto>
   - <ICMS>
      - <ICMS40>
           <orig>0</orig>
           <CST>50</CST>
        </ICMS40>
     </ICMS>
```

Figura 2. Tributação normal

Outra informação bastante importante é em relação à chave de acesso da nota fiscal eletrônica. Até a versão anterior a chave era composta de 44 dígitos numéricos sendo compostos da seguinte maneira, como mostra a **Tabela 1**.

Descrição	Caracteres
Código da Unidade Federativa	2
Data da emissão sendo composta de AAMM	4
C.N.P.J. do Emissor da NF-e	14
Modelo do Documento Eletrônico	2
Série do Documento Eletrônico	3
Número da NF-e	9
Código numérico	9
Digito Verificador	1
Totalizando	44

Tabela 1. Definição da chave de acesso na Versão do Schema 1.10

Agora com a mudança para este novo schema, a chave passou a compor 1 dígito a menos no código numérico e passou a ter o seguinte layout (**Tabela 2**).

Como é observado foi incluído mais um campo que é a forma de emissão da nota fiscal, e também, diminuído o campo de código numérico para 8 dígitos. Este campo de forma de emissão da NF-e é utilizado para definir se esta nota fiscal eletrônica foi emitida em ambiente normal ou em ambiente de contingência. Uma vez que é colocado que a nota foi emitida em contingência é obrigado a informação dos novos campos que no layout do schema 2.0 foram criados.

Descrição	Caracteres
Código da Unidade Federativa	2
Data da emissão sendo composta de AAMM	4
C.N.P.J. do Emissor da NF-e	14
Modelo do Documento Eletrônico	2
Série do Documento Eletrônico	3
Número da NF-e	9
Forma de Emissão da NF-e	1
Código numérico	8
Digito Verificador	1
Totalizando	44

Tabela 2. Definição da chave de acesso na Versão do Schema 2.0

São as duas novas tags do nó <IDE> que são: <dhCont> e <xJust>. Estas tags são para respectivamente informar a data/hora e a justificativa para a emissão daquela nota fiscal em específico no ambiente de contingência. Vale ressaltar que existem diversos tipos de contingência.

Outra alteração que foi adicionada é na parte dos itens da nota fiscal eletrônica (tag <prod>) que foi incluído uma tag <indTot> que faz o controle se o item em questão deve ser somado no total da nota fiscal, já que nesta nova versão há um efetivo controle se a soma do total dos itens da nota fiscal bate com o valor total da nota. Então esta tag faz a indicativa para que na validação ou na recepção dos dados pela receita federal, verifique se aquele determinado item deve ser somado ao total da nota ou não. Esta é uma regra que foi implementada na versão 2.0 e na 1.10 não se fazia este controle.

Uma das alterações solicitadas ao governo frente à nova versão é no que tange à integração entre sistemas. Como o arquivo XML é enviado ao governo e se autorizado em seguida é enviado ao destinatário da nota fiscal, mesmo antes da partida da mercadoria do estabelecimento emissor é possível que o destinatário possa saber de antemão o que foi faturado e está para ser entregue para sua empresa. Desta forma foi incluído na tag de itens também as tags xPed e nItemPed, que refere-se ao pedido de compra colocado pelo destinatário e qual é o item deste pedido de compra. Isso facilita muito a integração entre sistemas, podendo até ser considerada uma forma de troca de informações em meio eletrônico (EDI). Quem é um pouco mais velho e teve oportunidade de trabalhar com trocas de mensagens sabe que isso é uma prática comum entre empresas de diversos ramos, incluindo automobilísticos.

Vale ressaltar que cada uma das atividades específicas, como porte de arma, combustíveis, veículos zero km, tiveram alterações que podem ser observadas no manual do contribuinte. É interessante sempre a leitura e o estudo do mesmo para implementações desse formato da nota fiscal eletrônica.

Carta de correção eletrônica

A grande sacada do pessoal que está desenvolvendo a NF-e Brasileira é o fato de poder incluir eventos na nota fiscal eletrônica. Futuramente, quando houver uma operação interestadual, e a mercadoria passar através da fiscalização dos estados, será gerado um evento sobre aquela nota fiscal, mostrando que aquela nota passou por aquele posto de fiscalização em um determinado horário. Tudo isso será relatado através de eventos que serão mostrados na nota fiscal quando alguém efetuar a consulta de uma nota fiscal eletrônica.

Um dos primeiros eventos que o governo está liberando e que os contribuintes devem implementar é em relação à carta de correção. Apesar de ela estar prevista no projeto a um bom tempo atrás, é agora que ela foi preparada para que os contribuintes possam fazer os testes. Para os contribuintes que desejam fazer o teste basta entrar no site da NF-e e baixar os Schemas XML. Há também uma nota técnica que define como será o funcionamento do mesmo e a explanação dos campos existentes.

Implementando a nota fiscal eletrônica versão 2.0

Existem diversas formas para implementar a nota fiscal eletrônica no Delphi, seja ela usando componentes de terceiros, seja ela usando os Schemas dos xmls transformados em classes para poder fazer uso com o XMLDocument, dentre outras formas que podem se implementadas.

Uma das formas mais utilizadas hoje é fazer uso dos componentes ACBR/PCN. Estes componentes são de uso livre e podem ser copiados, alterados e redistribuídos sobre qualquer forma. A grande vantagem deste componente é que ele não é somente compilado para Delphi, mas tem um porte para o Lazarus também.

Para baixar estes pacotes basta fazer o download diretamente do repositório de dados (link na sessão de links) e compilar no Delphi XE para utilizá-lo. Este artigo irá demonstrar quais os passos e artimanhas para gerar esta nota fiscal eletrônica na sua versão 2.0 baseado neste componente, e desta forma, evitar todos os trâmites de encapsulamento de dados através do protocolo SOAP, que o componente já faz por padrão. Vale ressaltar que o ACBR é também uma suíte de componentes que pode ser utilizado para impressoras fiscais. Para o projeto SPED há um fork em desenvolvimento e também diversos outros itens que tentam automatizar ao máximo esse processo burocrático para o programador, fazendo com ele se preocupe somente com a regra do negócio propriamente dita.

Para fazer a instalação do componente no Delphi basta seguir os passos que estão no arquivo instalar, dentro do diretório de fontes na subpasta AcbrNfe2. Para continuar este artigo será tomado como base que o componente já esteja instalado no Delphi.

Iniciando o desenvolvimento com o AcbrNfe

Uma vez que há um projeto Win32 criado no Delphi, e com o componente AcbrNfe adicionado, para conseguir criar uma nota fiscal e transmiti-la, primeiramente precisa-se fazer a configuração do componente (**Figura 3**) para que desta forma ele possa transmitir para os Web Services do Estado correspondente. Para cada Estado há um Web Service que faz a recepção.

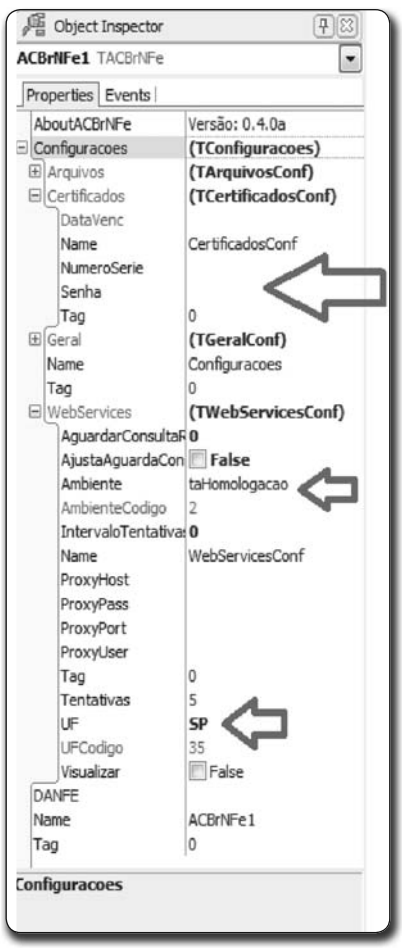


Figura 3. Configurações

As flechas indicam quais são as configurações básicas para que o componente possa trabalhar e enviar uma nota fiscal. Na primeira é necessário colocar o número de série do certificado digital, já que é ele que faz a assinatura digital do arquivo XML para poder ser enviado à receita federal. O campo senha é a senha do certificado utilizado para a assinatura do arquivo.

A segunda flecha indica qual é o ambiente que será enviado a nota fiscal em questão, é importante verificar sempre antes de enviar uma nota fiscal de teste, para não correr o risco de enviar para o ambiente normal podendo causar um risco para a empresa. É interessante ressaltar que independentemente de qual ambiente está trabalhando, os dois são idênticos, com a diferença de um ter validade jurídica (produção) e o outro não ter validade jurídica.

Listagem 1. Preenchimento dos campos para o envio da nota fiscal eletrônica

```
ACBrNFe1.NotasFiscais.Clear;
ACBrNFe1.NotasFiscais.Add;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.cUF := 35;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.cNF := 12345678;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.natOp := 'VENDA DENTRO DO ESTADO';
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.indPag := ipVista; // adicionar o uses em pcnConversao;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.modelo := 55;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.serie := 1;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.nNF := 1;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.dEmi := Date;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.dSaiEnt := Date;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.tpNF := tnSaida; // adicionar o uses em pcnConversao;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.cMunFG := 3531308;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.tpImp := tiRetrato; // adicionar o uses em pcnConversao;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.tpEmis := teNormal; // adicionar o uses em pcnConversao;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.tpAmb := ACBrNFe1.Configuracoes.WebServices.Ambiente;
// para pegar o ambiente configurado
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.finNFe := fnNormal; //adicionar o uses em pcnConversao;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.procEmi := peAplicativoContribuinte;
ACBrNFe1.NotasFiscais.Items[0].NFe.Ide.verProc := 'versão do aplicativo';
```

A terceira e última flecha indica para qual Web Service deve ir a nota fiscal eletrônica, no caso do exemplo ela está indo para São Paulo, vale lembrar que o Estado neste caso deve ser o estado de onde o contribuinte está cadastrado para emitir a NF-e.

Diante dessas configurações é possível iniciar o desenvolvimento da nota fiscal eletrônica em si. Na **Listagem 1** é apresentada a criação de uma nota fiscal de uma empresa fictícia.

O preenchimento dos campos para a geração da nota fiscal é simplesmente atribuir ao componente os valores desejados, para que depois ele possa gerar o xml. É bastante interessante verificar que existem algumas classes auxiliares como, por exemplo, a classe pcnConversao, que contém vários itens para facilitar a colocação de valores já pré-estabelecidos. É sempre recomendável que no início do preenchimento dos campos, caso se deseje preencher apenas uma nota fiscal, limpar as notas existentes no componente com a opção .notasfiscais.clear. Desta forma as notas são zeradas no componente podendo facilitar o trabalho.

Pode-se verificar também, se há mais de uma nota fiscal carregada no componente. Para poder acessar outras notas fiscais carregadas, basta informar qual é o item que se quer acessar, como no exemplo: acbrnfe1.notasfiscais.items[X], aonde o X representa o número de nota que se quer acessar no componente.

Uma vez que a nota fiscal está preenchida, é necessário fazer a assinatura digital deste arquivo, para que ele seja aceito juridicamente, comprovando desta forma a autenticidade do emissor e garantindo que esse documento seja válido eletronicamente e juridicamente. O componente disponibiliza um método para fazer a assinatura dos documentos que estão contidos no mesmo. No código a seguir consta o código que faz a assinatura das notas fiscais:

```
AcbrNfe1.Configuracoes.Certificados.GetCertificado;
ACBrNFe1.Configuracoes.Certificados.Senha := '123456';
ACBrNFe1.NotasFiscais.Assinar;
```

O método AcbrNfe1.Configuracoes.Certificados.GetCertificado, este método faz com que seja aberta uma janela aonde pode ser selecionado o certificado que está instalado na máquina. Desta forma é aberto uma lista de certificados (caso exista mais de um) e pode-se selecionar para que o sistema pegue o número de série do mesmo, fazendo com que o programador preencha somente a senha. Uma vez que esse certificado foi selecionado basta fazer a chamada ao método AcbrNFe1.NotasFiscais.Assinar para fazer a assinatura digital deste documento.

Após a assinatura da nota fiscal recomenda-se que seja feita a validação da mesma, para verificar se há alguns campos que não estão de acordo com o estabelecido no manual de orientação do contribuinte. Se existir algum item que não está em conformidade, é nesse momento que detectará a inconformidade. O componente da mesma forma que as outras ações, disponibiliza um método para a validação destas notas fiscais carregadas. O método em questão é o ACBrNFe1.NotasFiscais.Valida. Se existir algum erro em uma determinada nota fiscal, é gerada uma exceção mostrando qual é o erro que foi gerado. Se após gerar a exceção, for necessário capturar em qual nota está o erro, basta acessar a lista de alertas da nota que é disponibilizada através da propriedade alerta, podendo ser adquirida como no a seguir:

```
AcbrNfe1.NotasFiscais.items[X].Alertas
```

Lembrando que o caractere X representa o número da nota fiscal que deseja verificar os alertas. Depois que todos esses processos estão cumpridos é hora de enviar a nota fiscal para o webservice selecionado anteriormente. O componente ACBr faz o encapsulamento automático dos dados utilizando os requisitos necessários que são solicitados através do manual do contribuinte. Para fazer o envio desta nota fiscal basta chamar o método enviar. Para chamar este método basta utilizar o seguinte comando:

```
ACBRNFe1.WebServices.Envia(X);
```

Desta forma as notas fiscais presentes no componente serão enviadas para os Web Services da receita. O caractere X é o número do lote que está sendo enviado. Apesar do Web Service ser assíncrono, o componente fica durante um período fazendo consultas ao Web Service para conseguir pegar o resultado do envio da nota fiscal eletrônica, para saber se foi autorizada, rejeitada ou denegada. Se esse resultado vier de forma positiva, não é gerada nenhuma exceção no componente e desta forma, consegue-se gerar a nota fiscal eletrônica para ser armazenada e distribuída. Para salvar esta nota fiscal e poder distribuir, basta somente utilizar o seguinte método:

```
ACBrNFe1.NotasFiscais.Items[X].SaveToFile(LOCAL_ARQUIVO);
```

Novamente sendo X o número da nota fiscal no componente e local_arquivo é o local aonde o arquivo pode ser salvo. Uma vez que este arquivo está com a autorização de uso da receita federal já embutido no xml, neste momento ele se torna uma nota fiscal eletrônica que deve ser armazenada e enviada aos respectivos destinatários e também para a transportadora caso houver. Desta forma é feito a integração de envio do componente para com os servidores da receita federal.

Função de cancelamento

É muito comum dentro da regra de negócio de uma determinada empresa, o cancelamento de notas fiscais. No passado cancelavam-se as notas fiscais juntando as vias da nota fiscal e fazendo uma anotação de cancelamento. Com a nota fiscal eletrônica este procedimento passou a ser eletrônico também, e para isso é necessário que seja informado ao governo que uma determinada nota fiscal está sendo cancelada. Para este informe é utilizado, assim como a nota fiscal eletrônica, um arquivo xml com uma assinatura digital para conseguir cancelar. O cancelamento é bastante simples e com a utilização do componente ACBr se torna ainda mais fácil, já que toda a parte de encapsulamento o componente faz para quem deseja utilizar. O código a seguir mostra o cancelamento de uma nota fiscal:

```
AcbrNfe1.NotasFiscais.Clear;
ACBrNFe1.NotasFiscais.LoadFromFile(CaminhoArquivo);
AcbrNfe1.WebServices.Cancela('COLOCA-SE AQUI O MOTIVO DO CANCELAMENTO');
```

Basta carregar o arquivo XML da nota fiscal eletrônica no componente e fazer a chamada do método Cancelamento passando como parâmetro uma justificativa para o cancelamento da nota fiscal. Após este procedimento o componente AcbrNfe envia para a receita federal o arquivo xml solicitando o cancelamento desta nota fiscal, e em seguida já devolve um outro arquivo (de maneira transparente para o usuário) para verificar se o cancelamento foi aceito ou não. Para capturar os dados que são retornados do Web Service utiliza-se a classe do ACBrNfe que é a classe AcbrNFe1.WebServices.Cancelamento. Nesta classe estão os atributos preenchidos que vieram da resposta do Web Service.

Inutilização de números da nota fiscal eletrônica

O ACBrNfe disponibiliza também uma chamada nos métodos para fazer a inutilização de números da nota fiscal. Há algumas regras para utilizar esta funcionalidade da NFe. Mas resumindo, esta funcionalidade é utilizada quando um determinado número da sequência de números da nota fiscal é pulado. Por exemplo imagine que o próximo número a ser emitido é o 30, no entanto o sistema pula esse número e emite o 33. Deve-se então fazer a inutilização do número dos três números que foram saltados (30,31 e 32), já que não pode existir quebra de sequência na nota fiscal. Para fazer a chamada do método basta utilizar o seguinte método:

```
ACBrNFe1.WebServices.Inutiliza(CNPJ,
  Justificativa, ANO, MODELO, Série,
  NúmeroInicial, NúmeroFinal);
```

Na chamada do método Inutiliza são passados sete parâmetros que são: o Cnpj do emissor dos números que serão inutilizados, a justificativa informando o porque da inutilização, o ano, o modelo do documento que terá o seu número inutilizado, qual é a série que foi feita a quebra da sequência, o número inicial e final, que no caso citado seria o inicial sendo 30 e o final 33.

Após a chamada deste método o componente envia os dados para a receita, e fica aguardando a resposta do Web Service. Quando a resposta vem, pode-se acessar a resposta utilizando uma classe que é disponibilizada para a inutilização, que é a classe ACBrNFe1.WebServices.Inutilizacao. É nesta classe que estão todos os dados pertinentes a resposta do envio da solicitação de inutilização dos números da NFe.

Pegando a resposta o retorno do envio do xml da NFe

É muito comum o Web Service de recepção das notas fiscais eletrônica não conseguir entregar a mensagem em tempo hábil para o componente ACBrNfe. Apesar do Web Service ser um serviço assíncrono, o ACBrNfe quando faz o envio da nota fiscal, faz uma série de consultas repetidas para pegar a resposta do servidor. Na maioria dos casos isso é valido e a resposta vem quase que imediatamente. No entanto, em alguma situações o Web Service da receita demora mais do que o normal para enviar esta resposta e o componente não consegue capturar o retorno. Quando isso acontece é necessário fazer uma consulta posterior para pegar a autorização de uso. Para conseguir pegar esta resposta do servidor basta seguir o procedimento a seguir:

```
ACBrNFe1.NotasFiscais.Clear;
ACBrNFe1.NotasFiscais.LoadFromFile(CAMINHO_DO_XML_A_SER_CONSULTADO);
ACBrNFe1.Consultar;
ACBrNFe1.NotasFiscais.Items[0].SaveToFile(CAMINHO_A_SER_SALVO);
```

Para verificar todos os dados do retorno da consulta o ACBrNFe também disponibiliza uma classe para detalhar o retorno dos dados, esta classe é a ACBrNFe1.WebServices.Consulta. Acessando esta classe consegue-se apurar o status da nota fiscal, se está autorizada ou não e diversas informações que são pertinentes. No caso da linha que contém o método SaveToFile é neste momento que é gerada a nota fiscal eletrônica, já que é vinculada a autorização de uso com o arquivo XML enviado.

Impressão do DANFE

O Documento Auxiliar de Nota fiscal Eletrônica (DANFE) é um documento que é utilizado para fazer o acompanhamento da mercadoria no que tange o transporte. Ele tem validade se a nota fiscal estiver autorizada pelo governo, e poderá ser impressa em qualquer tipo de papel, que não seja papel de jornal. Há um modelo previamente estabelecido pelo governo e este pode ser visualizado no manual de integração do contribuinte.

Quando a empresa A emite uma nota fiscal para a empresa B e a empresa B não for emissora de nota fiscal eletrônica, a DANFE pode ser utilizada para ser arquivada como documento fiscal. Para as outras empresas a sua única função é o acompanhamento da mercadoria em si.

O componente ACBrNFe permite que seja emitido o DANFE. Para isso basta fazer a chamada de métodos específicos que serão demonstrados mais adiante. Um ponto bastante interessante é a emissão do DANFE no componente, não está presa somente em um único componente de impressão. Pode-se emitir o DANFE utilizando como base o Rave Report, o Fortes Report e até mesmo o QuickReport. São diversas opções que podem ser utilizadas. E isso, não irá variar como a chamada do método é feita. O que é necessário fazer é informar ao componente ACBrNFE qual é o componente que será responsável pela impressão da nota fiscal.

Para imprimir uma nota fiscal, basta fazer o que mostra o seguinte código:

```
ACBrNFe1.NotasFiscais.Clear;
ACBrNFe1.NotasFiscais.LoadFromFile(
  OpenDialog1.FileName);
ACBrNFe1.NotasFiscais.Imprimir;
```

Outro ponto que se destaca é que o componente também permite a impressão para um arquivo PDF. Para fazer a geração desse arquivo basta substituir o ACBrNFE1.NotasFiscais.Imprimir por ACBrNFE1.NotasFiscais.ImprimirPDF que será gerado um arquivo PDF contendo os dados do DANFE.

Enviando o email através do componente

No manual de integração do contribuinte que é emitido pela receita federal, fala que é necessário disponibilizar esta nota fiscal para o destinatário de alguma forma, seja através de e-mail, download através de um portal de arquivos xml ou de qualquer outra forma, no entanto, o destinatário da nota fiscal deve ter uma cópia deste arquivo XML também.

O componente ACBrNfe disponibiliza uma forma de enviar estas notas fiscais, que é o e-mail. Para isso basta informar ao componente qual é o servidor de e-mail que será utilizado e solicitar para que o mesmo faça o envio desta nota fiscal. O interessante de tudo isso é que se pode deixar todo esse processo integrado e transparente para o usuário que está operando o sistema de faturamento da empresa, tirando dele a responsabilidade por tais atos, evitando assim que a pessoa esqueça de enviar a nota fiscal a quem realmente deve recebê-la. O código a seguir mostra o envio do e-mail:

```
ACBrNFe1.NotasFiscais.Clear;
ACBrNFe1.NotasFiscais.LoadFromFile(
  CAMINHO);
ACBrNFe1.NotasFiscais.Items[0].EnviarEmail(ENDERECO_HOST,PORTA, USUÁRIO,
SENHA,USUARIO_EMISSOR, DESTINATARIO, ASSUNTO_EMAIL, MENSAGEM_EMAIL,
UTILIZA_SSL, ENVIA_PDF_JUNTO, COPIA_PARA, ANEXOS, CONFIRMA_LEITURA, AGUAR-
DA_ENVIO_EMAIL, NOME_REMETENTE, AUTO_TLS);
```

Carrega-se a nota fiscal no componente, e uma vez ela estando carregada, basta fazer o envio do e-mail passando os parâmetros necessários. Na **Tabela 3** estão a lista dos parâmetros do método enviarEmail.

São vários os parâmetros que devem ser configurados para fazer o envio do e-mail, no entanto, toda a regra de negócio está preparada para que o e-mail seja enviado, sem que seja necessário o programador ter que desenvolver o programa para fazer este procedimento. Com o envio do e-mail, completa-se o procedimento de envio da nota fiscal eletrônica.

Demais programas do AcbrNfe

Existem outros programas que fazem parte do projeto Acbr que auxiliam bastante a quem trabalha com nota fiscal eletrônica. É bastante comum, por exemplo, quando se recebe um arquivo XML e não consta o DANFE junto no email. No pacote de componentes do ACBr existe um componente que faz a impressão de um DANFE.

Número/nome Parametro	Descrição
1 – Endereco_host	Deve ser preenchido com o IP do servidor ou com o nome de rede válido
2 – Porta	Qual é a porta que roda o servidor SMTP
3 – Usuário	Usuário o qual deve ser feita a autenticação
4 – Senha	Senha para a autenticação
5 – Usuario_emissor	Geralmente é o mesmo que o parâmetro 3.
6 – Destinatario	E-mail de destino da nota fiscal eletrônica
7 – Assunto_Email	Assunto do e-mail
8 – Mensagem_Email	É o corpo da mensagem do e-mail
9 – Utiliza_SSL	Se irá utilizar criptografia SSL (True/False)
10 – Envia_PDF_Junto	Se irá uma cópia do DANFE junto com o e-mail
11 – Copia_para	É utilizado para enviar para outros e-mails
12 – Anexos	Se irá enviar mais algum anexo
13 – Confirma_leitura	Campo booleano que pede se solicita confirmação de leitura
14 – Aguarda_Envio_email	Se para o fluxo de execução e aguarda a totalidade do envio de email. Não é utilizado thread, se esta opção for verdadeira, o sistema ficará parado esperando o envio do e-mail.
15 – Nome_remetente	Nome do remetente do e-mail
16 – auto_TLS	Habilita ou desabilita a camada de transporte seguro

Tabela 3. Lista de parâmetros

Para trabalhar com esse programa é bastante simples basta arrastar um arquivo XML da NFe para ele, e automaticamente é aberto uma tela para se imprimir aquele XML na forma de representação textual que é o DANFE.

Outro projeto bastante interessante, principalmente para quem não tem tempo de desenvolver para a nota fiscal eletrônica, é utilizar o programa Monitor. Este programa funciona ou por meio de arquivos textos, ou por sockets. O monitor fica rodando no systray e fica aguardando comandos para fazer executar os processos solicitados. Existe inclusive um arquivo de help que mostra como executar os comandos, e de que forma o monitor irá mostrar o resultado.

O programa RecuperaXML que também vem junto, é um utilitário para conferir e conseguir pegar o xml enviado para a nota fiscal. No entanto este programa deve ser utilizado com bastante cuidado, já que ele utiliza e resgata dados da consulta disponível pela internet que pode ser acessada via browser, então qualquer alteração na página pode fazer com o que o programa apresente dados errados.

Conclusão

A nota fiscal eletrônica é uma realidade irreversível, desta forma a grande maioria dos contribuintes terão uma hora que implementá-la. No entanto, já existem várias regras de negócio que os programadores devem se preocupar para conseguir colocar uma solução eficaz para o cliente. É interessante então, utilizar pacotes de componentes que são opensource para automatizar e agilizar os processos que são técnicos.

Contudo, mesmo utilizando esses componente, temos alguns macetes para conseguir integrar ao sistema. Utilizando o ACBrNfe que é um componente bastante conceituado e testado pela comunidade, evita-se transtornos de erros, além claro, de que qualquer dúvida a comunidade está aí para auxiliar, já que é bastante ativa.

Entendendo esse procedimento de envio, nada impede de fazer um sistema de entrada, aonde o mesmo possa alimentar o sistema automaticamente, já que isso é possível utilizando o ACBrNfe também, fica aí a sugestão.

Jederson Zuchi
jederson@sygnux.com.br
É diretor de sistemas industriais da Sygnux Software (www.sygnux.com.br) localizada em Monte Alto/SP. Formado pela Fatec/TQ, desenvolve com Delphi desde a versão 3. Atualmente trabalhando com Delphi, Java e Plataforma .NET em diversos ERPs industriais. Professor do SENAC/SP e da Fatec/TQ lecionando em diversas matérias de desenvolvimento de software. Twitter @jedersonzuchi.

Site da Nota Fiscal Eletrônica
http://www.nfe.fazenda.gov.br

Diversos links para baixar o acbr
http://acbr.sourceforge.net/drupal/?q=node/11

Blog do autor
http://twitter.com/jedersonzuchi

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

FlameRobin

Conheça esta ferramenta gratuita para administração de BD Firebird

Fabrício Hissao Kawata

Resumo DevMan

De que trata o artigo:

Este artigo tem como finalidade apresentar ao leitor a ferramenta open-source FlameRobin, voltada exclusivamente para a administração e manipulação de bancos de dados Firebird. Além das características básicas que toda ferramenta deste tipo deve ter, o FlameRobin ainda possui um caráter multiplataforma, podendo ser utilizado nativamente em sistemas operacionais Windows, Linux, FreeBSD, Solaris e MacOS X.

Em que situação o tema é útil:

O FlameRobin proporciona a seus usuários uma forma bem intuitiva de administrar bases de dados Firebird, através de recursos totalmente visuais. Por via deles, é possível facilmente criar e manter os principais objetos de um banco de dados, tais como tabelas, views, stored procedures, triggers, entre outros. Além disso, atividades mais específicas como, por exemplo, backup e restore, também são proporcionados pela ferramenta.

FlameRobin:

O artigo terá enfoque na apresentação da ferramenta FlameRobin ao leitor, visando demonstrar alguns de seus principais recursos, tão importantes e essenciais na rotina diária de administração de bancos de dados Firebird. Por ter sido concebido com o objetivo de ser o mais simples e eficiente possível, o FlameRobin tende a ser muito bem aceito pela comunidade de desenvolvedores Delphi, em vista do casamento perfeito das aplicações Win32 com acesso a banco de dados com o SGBD Firebird e consequentemente pela necessidade de administrá-lo devidamente.

O Firebird é um dos SGBDs (Sistema Gerenciador de Banco de Dados) mais famosos em meio à comunidade Delphi, visto sua grande utilização no desenvolvimento de aplicações Desktop que envolvem acesso a banco de dados. Entre os motivos da sua boa aceitação por parte dos desenvolvedores, está sua simplicidade e praticidade de uso, seu alto desempenho e principalmente seu caráter open-source e gratuito. Em vista desses fatores, há um feedback muito grande em torno do tema na internet, onde comunidades e grupos de discussão foram formados, facilitando assim a obtenção de apoio e ajuda técnica de diversas fontes.

Conceitualmente falando, o projeto Firebird é coordenado e mantido pela Fundação FirebirdSQL, uma empresa sem fins lucrativos que foi criada para centralizar o avanço do projeto como um todo. Além disso, o Firebird tem todo seu código-fonte aberto e disponibilizado na rede Web a todos que venham a ter algum tipo de interesse (seja para contribuição, testes ou apenas curiosidade). Ainda falando em código, o Firebird teve sua base iniciada em meados do ano 2000, a partir do código de outro SGBD, mais precisamente do então Borland Interbase 6, que era o nome do produto na época (atualmente o Interbase faz parte do quadro de produtos da Embarcadero).

Uma curiosidade: o release 1.0 do Firebird é fruto do resultado obtido pelo trabalho de um grupo de programadores que, em conjunto, uniram-se para corrigir algumas inconsistências e problemas detectados no Interbase. Todo esse trabalho culminou num produto único, de características próprias e que hoje já caminha por suas próprias pernas. Atualmente (até o fechamento deste artigo), o Firebird se encontra na versão 2.5, que é sua distribuição estável mais recente.

FlameRobin

O FlameRobin é uma ferramenta que proporciona um ambiente totalmente visual voltado exclusivamente para a administração de servidores Firebird, mantendo suas bases de dados, bem como a manipulação de seus dados. Assim como o próprio Firebird, o FlameRobin possui dentre suas características principais o caráter gratuito, open-source e multiplataforma, podendo ser devidamente utilizado em ambientes Windows, Linux, FreeBSD, Solaris e MacOS X. Por ser de código aberto, essa lista poderá ser aumentada com o passar do tempo, de acordo com a necessidade da comunidade de seus utilizadores mais fiéis.

Seu principal autor é um desenvolvedor europeu chamado Milan Babuskov, que em

meados 2003 deu início a um projeto próprio que visava a construção de uma ferramenta totalmente gráfica, leve e que rodasse em plataformas diversas, voltada à administração de bancos Firebird. Neste trajeto, Milan utilizou-se de ferramentas e bibliotecas open-source, inclusive mantendo o produto final desta mesma forma, ao mesmo tempo em que nomeava o então projeto como FlameRobin. Nome este, aliás, que assim como sua inspiração (o Firebird – pássaro de fogo), também faz referência a um pássaro, o Flame Robin, uma ave nativa da Austrália.

Atualmente, com a ferramenta já estabilizada, seus autores ainda lhe atribuem alguns objetivos, que de certa forma servem de guia ao projeto:

• Ser uma forte ferramenta GUI: a grosso modo, o termo GUI (Graphical User Interface), em termos de software, é um tipo de interface de usuário que permite a interação do utilizador com o aplicativo através de imagens, ao invés de comandos de texto. Segundo os autores, o primeiro objetivo do FlameRobin é ser para seus usuários (principalmente aqueles que estão iniciando com o Firebird) uma robusta ferramenta GUI, de forma a obstruir qualquer uso de linhas de comando para a manipulação das bases de dados;

• Ser uma tendência Linux: outro objetivo dos autores é estabilizar o FlameRobin como uma ferramenta GUI destinada principalmente para a plataforma Linux. Apesar de soar estranho num primeiro momento, a justificativa dos autores para isto é de que, atualmente, as opções de ferramentas GUI, com os mesmos propósitos do FlameRobin, para Windows são inúmeras (sejam elas gratuitas ou pagas), diferente da realidade no Linux, onde há uma escassez;

• Ser parte integrante do Firebird: finalmente, outro desejo dos autores do FlameRobin é disponibilizar uma versão enxuta (lite) da ferramenta, juntamente do servidor Firebird, compondo assim um "kit de primeira viagem" para os novos usuários do SGBD.

Baixando e instalando

Como já mencionado, o FlameRobin é gratuito e pode ser baixado diretamente de seu site oficial (ver seção Links). Conforme seu foco principal, de atingir um alto nível de simplicidade, a ferramenta torna isso perceptível pelo tamanho mínimo de seu instalador, inferior a 2MB. Outro ponto a ser observado é relativo à própria instalação, que se utiliza do famoso padrão "Next-Next-Finish" através de um Wizard bem intuitivo (**Figura 1**).

Lembrando que a situação apresentada é para ambientes Windows, que é a realidade da maioria dos desenvolvedores Delphi, portanto, numa eventual instalação em outros ambientes, pode-se obter detalhes no próprio site do projeto. Continuando, outro detalhe em relação à instalação, é que por se tratar de uma ferramenta exclusiva ao Firebird, se torna incondicional a presença da dll cliente deste, na máquina onde está sendo instalado o FlameRobin. Conforme orientações presente em seu próprio site oficial, o arquivo gds32.dll é o recomendado para a versão 1.0 do Firebird, enquanto que para suas versões 1.5, 2.0 e superiores, a fbclient.dll é a ideal.

Nota do DevMan

SGBD é a sigla para Sistema de Gerenciamento de Banco de Dados, que vem do termo inglês Database Management System (DBMS). Pode ser definido como um conjunto de softwares que interagem entre si, provendo o gerenciamento de uma base de dados. Entenda por gerenciar, o processo de modificar, persistir, armazenar e obter informações de determinado banco de dados. Visando tirar a responsabilidade das partes clientes, o SGBD oferece a estas uma interface formada por APIs (Application Programming Interface) e drivers exclusivos que provêem a execução de comandos na linguagem SQL (Structured Query Language). São exemplos de SGBDs já consagrados: Firebird, MySQL, Oracle, MS SQL Server, PostgreSQL, entre outros.

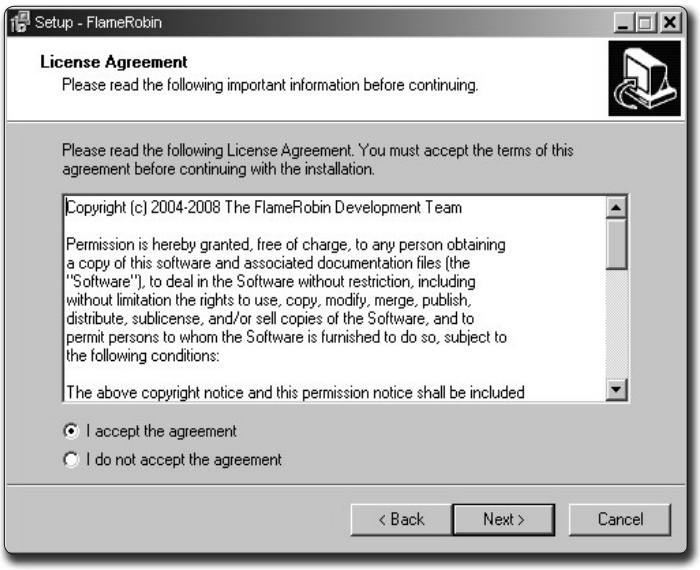


Figura 1. Wizard de instalação

Registrando e acessando um banco de dados existente

Uma vez instalada, a ferramenta já está apta a ser utilizada. Sua interface é apresentada na **Figura 2**. Logo, o primeiro passo é fazer o registro de um servidor Firebird válido. Esta opção está disponível através de seu item de menu denominado intuitivamente como Server. Neste momento será necessário informar o nome do Host do servidor (Ex.: em um ambiente local, é de praxe usar localhost, já em ambientes remotos, geralmente usa-se o IP do servidor) e o número da porta utilizada para a conexão. Com o Server registrado, o FlameRobin já pode estabelecer conexão com as bases de dados atreladas a este servidor, bem como criar novas bases.

Na **Figura 3** temos o exemplo de um banco de dados existente, mais precisamente o banco Employee que acompanha a instalação padrão do Firebird, sendo registrado na ferramenta. Recurso este que está disponível no menu Database sob o nome de "Register existing database...". Ao selecionar esta opção, uma nova janela é exibida para que sejam informadas as credenciais da base a ser registrada. Contudo, essas são informações básicas e que todo desenvolvedor precisa utilizar neste tipo de ferramenta, tal como o caminho do banco e os dados de autenticação (usuário e senha). Ainda sobre autenticação,

a ferramenta oferece algumas opções pré-definidas de acesso ao banco de dados, que podem ser escolhidas dentre os itens relacionados a Authentication.

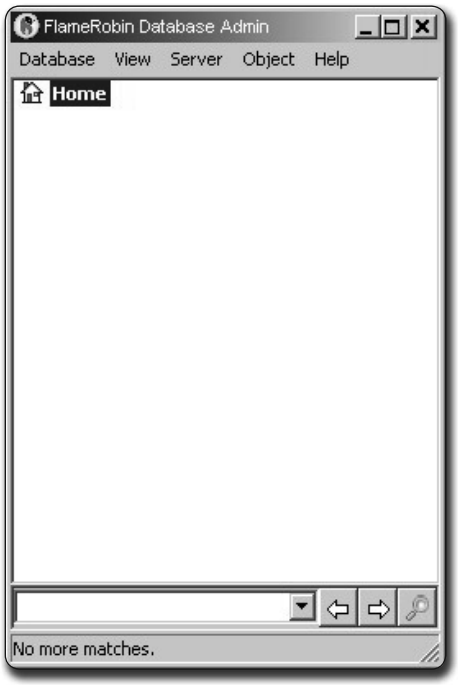


Figura 2. Interface do FlameRobin

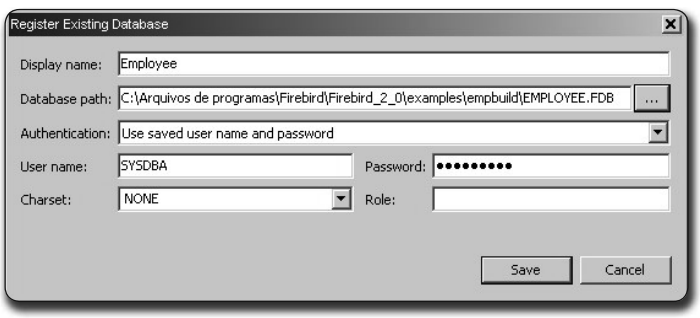


Figura 3. Registrando um banco de dados existente

Só para exemplificar, a opção "Use saved user name and password" salva as informações de usuário e senha informada, e as utilizam sempre que uma conexão com o banco for solicitada. Já para "Use saved user name, but always enter password", sempre será necessário informar a senha, para se conectar ao banco. Com a base de dados registrada, basta conectar-se a ela, utilizando a opção Connect também do menu Database. Estando tudo correto, a estrutura da base de dados é disposta na área principal do FlameRobin, ou seja, todos os seus objetos são mostrados, tais como: tabelas, trigger, views, procedures, domains, entre outros, conforme pode ser conferido na **Figura 4**.

Tecnicamente falando, no momento em que a opção Connect é acionada, o FlameRobin faz a carga parcial das informações de metadados da base de dados de modo que seja satisfatória num primeiro momento. Dessa forma obtém-se uma alta velocidade por parte da ferramenta, no momento da conexão com o banco. Conforme a navegação pelos objetos ocorre, as demais informações de metadados são recuperadas por demanda.

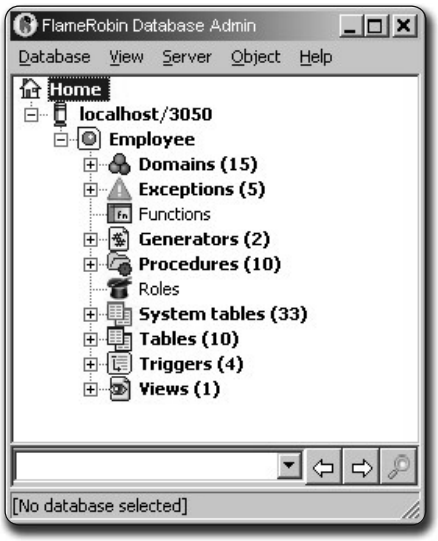


Figura 4. Estrutura do banco Employee sendo devidamente mostrada no FlameRobin

Criando um banco de dados

Visto como registrar um banco de dados existente na interface do FlameRobin, outra atividade corriqueira de todo desenvolvedor é a criação de uma base de dados. Como exemplo, tomemos as situações onde é necessário criar um novo banco para fins de testes, ou principalmente para um novo projeto que se inicia. Neste caso, a ferramenta oferece este recurso novamente pelo menu Database através do item "Create new database...". Novamente, neste tipo de processo é imprescindível informar alguns dados como o caminho do banco que será criado, as credenciais de autenticação (usuário e senha), que são idênticas às citadas anteriormente. O diferencial aqui fica por conta das informações de Page Size e SQL Dialect, conforme visto na **Figura 5**. Esses dados são de suma importância neste momento, pois irão definir o comportamento estrutural e de dados do novo banco. Page Size se refere ao tamanho da página de armazenamento dos registros enquanto que SQL Dialect define a forma de interpretação de determinados tipos de dados que serão usados pelas colunas das tabelas do banco de dados.

Executando scripts e instruções SQL

Como toda ferramenta deste tipo, o FlameRobin possibilita a execução de scripts e instruções na linguagem SQL, de forma a permitir a realização de atividades como consultar dados de tabelas, criar objetos, alterar sua estrutura, entre outros. Para tal, a ferramenta designa uma janela exclusiva para este recurso, que pode ser acessado pelo menu Database, através da opção "Run a query...". Conforme pode ser visto na **Figura 6**, esta janela trata-se de um pequeno editor de código, com as funcionalidades de auto completar, execução, salvamento e execução do script ou código inserido no editor. Como será

visto adiante, este mesmo editor serve de base para outras funcionalidades da ferramenta.

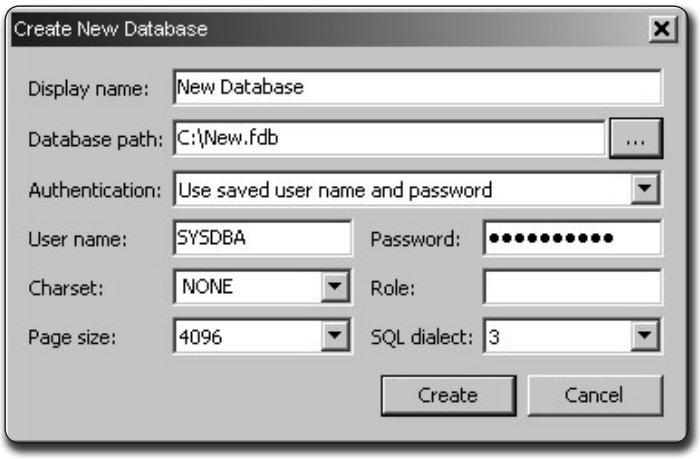


Figura 5. Criando um novo banco de dados

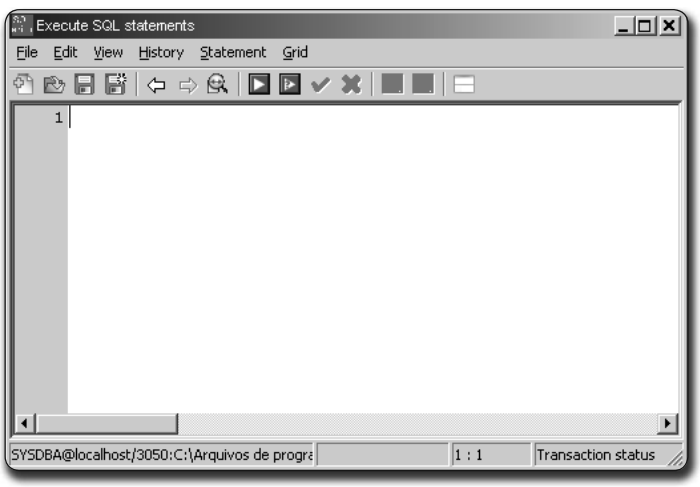


Figura 6. Editor para execução de código SQL

Manipulando tabelas

Uma tabela em um banco de dados relacional, como é o Firebird, é considerada uma de suas principais peças. Caracteriza-se por ser uma estrutura que armazena uma porção de dados dispostos em linhas e colunas. Neste caso, considera-se cada linha como sendo um bloco de informações relacionadas às colunas, as quais representam os campos da tabela. Em vista disso, o FlameRobin oferece recursos para a manipulação de tabelas, como a criação, alteração e manutenção de seus dados.

Presente no menu Object, a opção New Table realiza o processo de criação de uma nova tabela. A ferramenta faz chamada ao editor SQL, já com todo o código previamente montado para a construção do novo objeto. Uma vez criada, a tabela fica disponível na estrutura treeview da sua base de dados, para as atividades subsequentes. Dentre estas está o manuseio dos seus dados. No FlameRobin, isto é realizado pela opção "Insert into..." do mesmo menu. Quando acionado, este item novamente abre o editor de código, porém desta vez pré-montado com a instrução INSERT INTO, direcionada para a adição de dados em uma tabela. Executando o comando, as informações são persistidas no banco e ficam disponíveis para visualização.

Para se poder ver os dados de uma tabela, o FlameRobin traz um recurso intuitivamente denominado como "Select from...". Como o próprio nome sugere, uma instrução SELECT será executada sobre a tabela, retornando os seus dados dispostos em uma grade visual, conforme mostra a **Figura 7**. Nesta mesma janela, ainda é possível, além da visualização, realizar as operações básicas sobre os dados como: inclusão, exclusão e alteração. Os dois primeiros ficam disponibilizados na forma de botão na toolbar da janela, enquanto que a edição pode ser feita diretamente no grid de dados. Mencionando que toda e qualquer atividade deste tipo, para ser efetivada, necessita ser confirmada com uma ação de Commit (botão com um símbolo de visto). Da mesma forma, para se reverter uma atividade, é necessária uma ação de Rollback, acionado pelo botão com o símbolo X.

Ainda sobre estas operações, cada uma delas gera uma instrução SQL que é executada "por baixo dos panos" pelo FlameRobin. No entanto, todo este processo pode ser acompanhado na aba Statistics, onde são exibidas todas as instruções executadas pela ferramenta, bem como seu plano de execução, tempo, dentre outras informações analíticas.

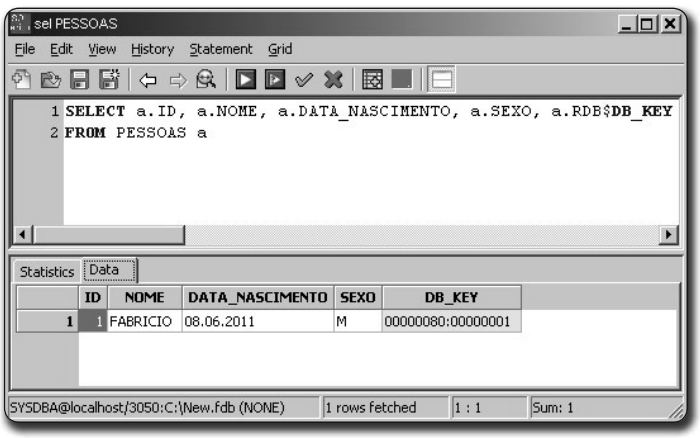


Figura 7. Exibindo os registros de uma tabela

Constraints

Constraints são restrições aplicadas às colunas de uma tabela, sendo essenciais no contexto de um banco de dados. Em via de regra, citando especificamente o Firebird, as constraints possuem diferentes tipos, de acordo com a função que exercem:

- Primary Key: define a(s) coluna(s) que formarão a chave primária da tabela, de forma a criar um índice único perante a estrutura desta;
- Foreign Key: estabelece a(s) coluna(s) cujas informações estão relacionadas a outras tabelas da base de dados;
- Unique: define a(s) coluna(s), cujos valores serão únicos, ou seja, não poderá haver mais de um registro com o mesmo valor para os campos definidos;

• Check: determina uma condição (checagem), que deverá ser respeitada para que as informações sejam salvas na tabela.

O FlameRobin disponibiliza recursos para que todos estes tipos apresentados sejam facilmente manuseados. Na prática, isto é conseguido acessando-se a opção Show Properties do menu Object, que inicialmente exibe um resumo de toda a estrutura da tabela. Dentre suas abas de opções, há um item voltado especificamente a Constraints.

Na **Figura 8** é mostrada esta funcionalidade da ferramenta. Para definir, por exemplo, uma chave primária para uma tabela, é necessário simplesmente acionar a opção Add primary key para que a própria ferramenta se encarregue de exigir o fornecimento das informações para a conclusão do processo, tal como o nome da nova constraint Primary Key e a escolha do(s) campo(s) da tabela que ficará associado. Novamente, o que o FlameRobin faz é gerar e executar de forma automática o código SQL necessário para a criação da constraint.

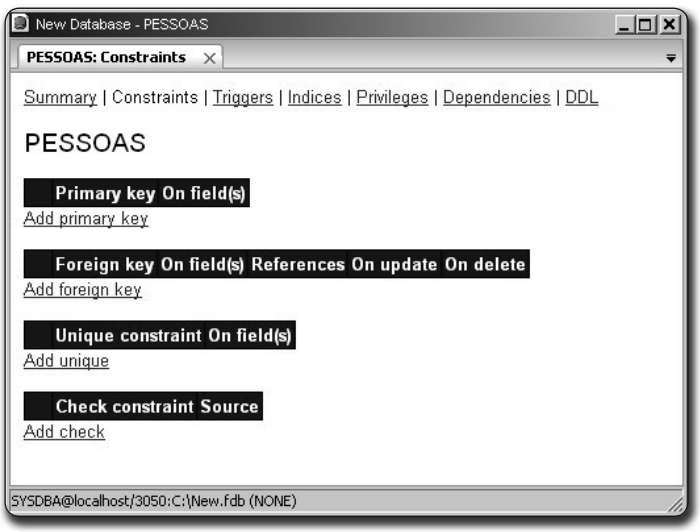


Figura 8. Opções de constraints de uma tabela

Novamente o editor é chamado, conforme pode ser visto na **Figura 9**, exibindo na sua parte inferior o código que foi executado, restando apenas ao usuário confirmar a operação, através de uma ação de Commit.

Trabalhando com Views

Uma view pode ser definida como uma tabela virtual dinâmica, armazenada no banco de dados, cujos dados são fruto dos resultados obtidos de consultas sobre outras tabelas físicas. Dessa forma, você pode ter uma estrutura pré-pronta de dados correlacionados entre várias tabelas e acessível como se fosse uma simples tabela. Ainda sobre as vantagens em se ter este tipo de objeto, está a economia de escrita de código, que se faz necessária apenas uma única vez, ficando a instrução armazenada e pronta para uso a qualquer momento. Outro ponto de ressalva é a possibilidade de compor e nomear as colunas de uma view de forma livre, podendo utilizar nomes mais amigáveis e combinações de outros campos de outras tabelas e assim encapsular toda a complexidade envolvida.

No FlameRobin, tanto a criação quanto a manutenção de views se dá de maneira semelhante às tabelas. Uma diferença fundamental é percebida no momento da visualização de sua estrutura, onde, além dos campos, é mostrado também o código formador da view, ou seja, a instrução SELECT. Na **Figura 10** pode ser visto a estrutura da view PHONE_LIST do banco de dados Employee do Firebird. Na seção Source, é exibido o comando SELECT que dá origem às informações contidas nesta view, onde há o envolvimento de duas tabelas distintas, que resultam numa estrutura única.

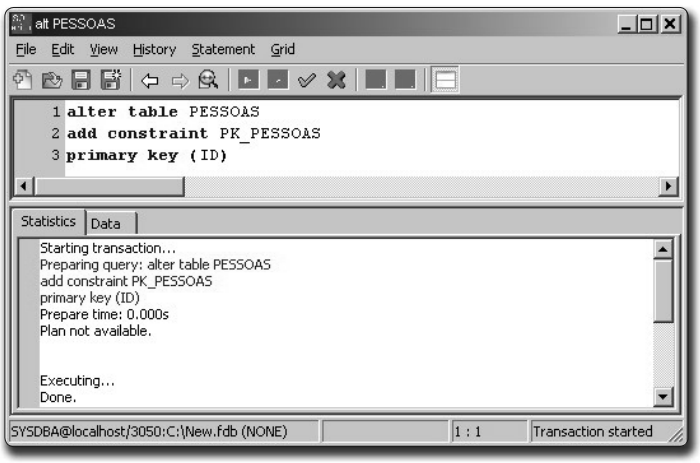


Figura 9. Criando uma constraint Primary Key para a tabela

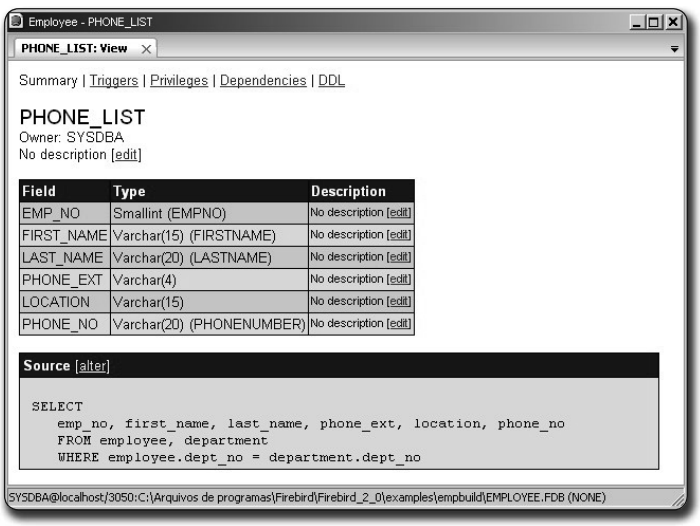


Figura 10. Estrutura da view PHONE_LIST

Trabalhando com Stored Procedures

Stored Procedures são procedimentos armazenados em um banco de dados, que encapsulam uma coleção de comandos na linguagem SQL, que interagem num processo único em função de um objetivo de negócio. Além disso, as Stored Procedures contribuem para a diminuição do tráfego na rede e para a otimização de desempenho na interação com o banco de dados, visto que todo seu processamento é realizado no próprio servidor.

Assim como acontece com os demais tipos de objetos do banco de dados, a criação de uma Stored Procedure no FlameRobin envolve o menu Object, através do item New Procedure. Novamente a ferramenta já apresenta um script previamente produzido do corpo do objeto, mostrado na **Listagem 1**.

Listagem 1. Código produzido para criação de Stored Procedure

```
SET TERM ^ ;

CREATE PROCEDURE name
 ( input_parameter_name < datatype>, ... )
RETURNS
 ( output_parameter_name < datatype>, ... )
AS
DECLARE VARIABLE variable_name < datatype>;
BEGIN
  /* write your code here */
END^

SET TERM ; ^
```

Pela listagem apresentada, pode-se notar que o FlameRobin já faz toda a indicação dos locais onde deverão ser informados alguns dados e a codificação do procedimento em questão. Para um melhor entendimento, deixe em letras maiúsculas as palavras relativas à SQL e em caixa baixa as informações que servem somente de guia para o utilizador. Dessa forma, fica fácil presumir que, por exemplo, em "name", será necessário atribuir um nome a Procedure e em "input_parameter_name" e "output_parameter_name" declarar os possíveis parâmetros de entrada e saída que a Stored Procedure venha a ter.

Backup

O termo backup, no contexto geral da informática, consiste na cópia dos dados de um local de armazenamento para outro, para que sejam restaurados a qualquer momento, numa eventual necessidade, como quando da perda, por qualquer motivo, dos dados originais. Passando este conceito para o ambiente de banco de dados, esta "cópia de segurança" abrange não só os dados, como também toda a estrutura dos objetos envolvidos na base que será copiada. Dessa forma, esta simples tarefa torna-se de extrema necessidade no ambiente de qualquer desenvolvedor.

No caso do Firebird, este processo pode ser feito nativamente utilizando-se um utilitário de linha de comando denominado "gbak". Entretanto, adotando-se o FlameRobin como ferramenta de apoio para a manutenção de um banco, o processo de backup pode ser realizado facilmente de forma visual. Isto se dá através da opção Backup database, presente no menu Database. Vale ressaltar que esta operação exige que um dos bancos de dados registrados na interface esteja selecionado, pois ele será a fonte do processo. Na **Figura 11**, a janela do recurso é mostrada. Fica claro todo o encapsulamento que o FlameRobin faz das informações do gbak para a realização do backup, proporcionando uma interface intuitiva e simples de usar. Em vista disto, somente é necessário informar o local e o nome do arquivo a ser gerado, além de marcar as opções de backup que convenha:

- Ignore checksums: Esta opção força a realização do backup, mesmo quando houver problemas nas páginas do banco de dados. Uma vez desmarcada esta opção, caso erros sejam detectados, o processo de backup é abortado;
- Ignore limbo transactions: Quando marcado, Ignore limbo transactions descarta toda e qualquer transação corrompida presente na base de dados;
- Use non-transportable format: Esta opção salva os dados e metadados do banco em um formato voltado para a versão atual do servidor de banco de dados, restringindo sua restauração em outras diferentes versões;
- Only backup metadata: Produz uma cópia vazia do banco, ou seja, efetua a cópia somente de sua estrutura, descartando os dados;
- Don't perform garbage collection: Deixa de executar o Garbage Collection do Firebird, recurso este que faz a limpeza de todo e qualquer lixo presente no banco, bem como a reorganização de seu espaço de memória utilizado. Isso faz com que o processo de Backup seja realizado de forma mais rápida;
- Convert external tables: Converte tabelas de arquivos externos para tabelas internas no banco de dados.

O início do processo se dá pelo botão Start Backup e a caixa "Show complete log", quando marcada, exibe o log de toda a operação que está sendo realizada no momento da execução do processo.

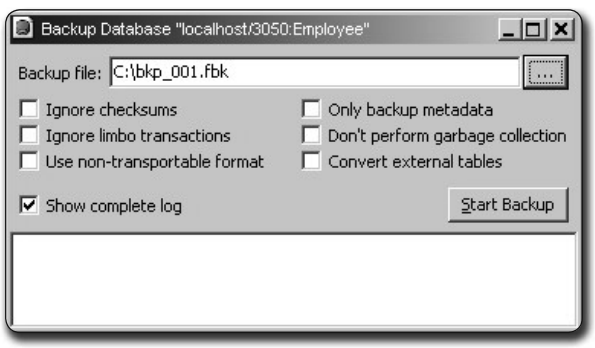


Figura 11. Fazendo Backup da base Employee

Restore

O Restore pode ser considerado o processo conseguinte ao Backup, visto que, sua ação é a de restaurar os dados armazenados na cópia de segurança. Em termos de banco de dados, mais especificamente no Firebird, um Restore se faz necessário, por exemplo, nas seguintes situações:

- O Firebird estabelece uma limitação de 255 vezes, relativo ao número de alterações dos metadados de uma tabela. Uma vez atingido este limite, novas alterações não são permitidas, sendo necessária uma operação de Restore para que esse número limite seja zerado novamente;
- Quando for necessário alterar o tamanho de página (Page Size) do banco de dados;
- No caso de mudança da versão do Firebird (por exemplo, a migração de um banco na versão 1.5 para a mais recente, 2.5),

torna-se imprescindível o Backup na versão antiga e a seguir o Restore na nova versão.

Ainda no Firebird, para se restaurar o backup de um banco de dados, novamente usa-se nativamente o utilitário gbak, e novamente o FlameRobin encapsula todo este processo, provendo recursos visuais para a ação de Restore. Esta opção é disponibilizada pela ferramenta também no menu Database, através do item Restore database. Na **Figura 12** é mostrada sua janela no momento da realização de um Restore. No rodapé, onde fica a amostragem dos logs do processo, pode ser visto o uso do gbak pela própria ferramenta, que se dá de modo transparente ao usuário, afinal, todas as informações necessárias já foram encapsuladas, resultando nas opções visuais disponíveis na parte superior da janela.

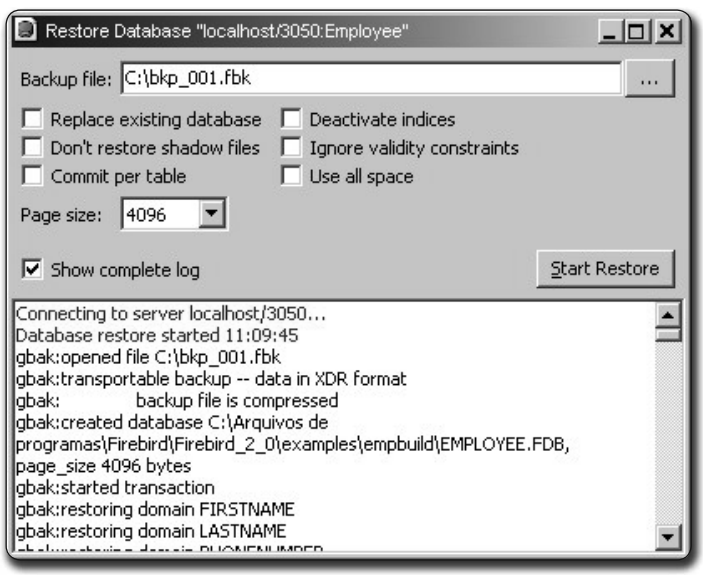


Figura 12. Fazendo o Restore de um backup

Nesta janela também estão dispostas as opções possíveis de Restore, da mesma forma que é mostrada no recurso de Backup:
• Replace existing database: Esta opção faz com que o processo de Restore substitua o banco de dados existente pelo que está sendo restaurado;
• Don't restore shadow files: Com esta opção marcada, os arquivos de sombra (shadow files) não são restaurados. Um arquivo shadow é uma cópia adicional do arquivo de banco de dados, sendo que para cada banco podem existir vários arquivos destes;
• Commit per table: Por padrão, o Firebird restaura primeiramente todos os metadados, para em seguida restaurar os dados. Com esta opção marcada, para cada tabela, seus metadados e dados serão restaurados em sequência e, logo em seguida, um Commit será aplicado. Caso não haja inconsistências nos dados, a operação é confirmada. Dessa forma, este tipo de abordagem pode ser útil para se detectar possíveis corrupções em um arquivo de Backup que está sendo restaurado, podendo mais facilmente apontar onde se encontra o problema;
• Deactivate indices: Esta opção desativa os índices presentes no banco de dados durante o Restore, aumentando a velocidade do processo. Isto pode ser explicado pelo fato de que o Firebird primeiro restaura todas as tabelas e seus dados, para só então atualizar seus índices;
• Ignore validity constraints: Uma vez marcada, esta opção faz com que as constraints relacionadas a tabelas e campos não sejam restauradas. Fazendo menção a uma situação de exemplo, é possível mensurar a utilidade desta opção. Suponha que uma determinada tabela possua uma constraint e que esta validação sofra alguma alteração depois de vários dados já terem sido inseridos nesta tabela. Dependendo do tipo de verificação que é feita pela nova regra da constraint, um erro poderia ser levantado durante o processo de Restore, em vista dos dados que já haviam sido lançados antes da mudança e que se tornaram inconsistentes depois dela;
• Use all space: Por padrão, um banco Firebird preenche suas páginas até 80% do limite de sua capacidade. Use all space faz com que este espaço seja preenchido por completo.

Gerando script DDL

DDL, ou Data Definition Language, é a linguagem de definição de estruturas de dados. Em suma, um simples script na linguagem SQL que cria uma tabela ou qualquer outro objeto (até mesmo o próprio banco de dados), é um script DDL. No ambiente real, a utilização destes tipos de scripts é essencial, por exemplo, nos casos onde seja necessário gerar uma réplica da estrutura de determinado objeto em outro banco.

No FlameRobin, é possível gerar um script DLL do banco por completo, ou das partes (objetos) que lhe convenha. Para o primeiro caso, usa-se a opção "Extract metadata DDL...", presente no menu Database. Mediante isso, o editor SQL é aberto, já com todo o script disponível referente à estrutura geral do banco selecionado. Já para se obter a DDL de um objeto específico, uma opção exclusiva para este fim (denominada DDL) pode ser encontrada em sua janela de propriedades, que é acessada através do menu de contexto do objeto desejado. Na **Figura 13** é mostrada a DDL de um Generator do banco Employee do Firebird, sendo visualizada na janela de propriedades do objeto. Nos casos onde seja necessário salvar ou executar tal script, utiliza-se a opção "open in SQL editor", para que o código seja aberto no editor de código do FlameRobin.

Manuseando privilégios

Em bancos de dados Firebird, cada um de seus objetos possui uma lista de privilégios que identificam os usuários e as ações que estes podem executar sobre o objeto. As ações em questão são cinco: Select, Insert, Update, Delete e Reference.

No FlameRobin, a visualização e manutenção destas opções perante um objeto, se dá novamente através de sua janela de propriedades, pela opção denominada Privileges (**Figura 14**). Ainda com base na imagem, a concessão ou revogação de privilégios se dá pelo link "Grant and revoke privileges", que abre uma nova janela onde é possível manusear não só este objeto, como todos os demais. Ao final do processo, o editor

SQL novamente é chamado, já com as devidas instruções SQL de GRANT (conceder) ou REVOKE (revogar) dos privilégios.

Figura 13. Visualização da DDL de um objeto

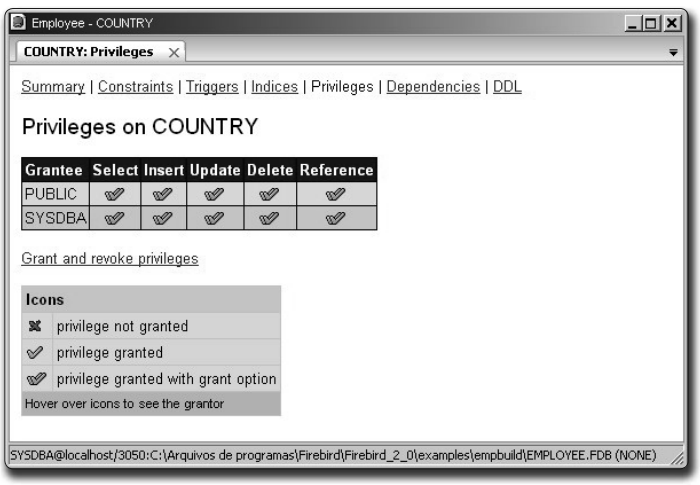


Figura 14. Visualização dos privilégios da tabela Country, banco Employee

Gerando dados testes

O FlameRobin traz ainda outro recurso interessante que é a capacidade de gerar dados com valores fictícios para uma tabela, com a finalidade de execução de testes. Dito isto, todo desenvolvedor pode imaginar o quão produtivo isto pode se tornar e facilitar seus testes de carga e funcionalidades em sistemas que acessam banco de dados Firebird.

Na ferramenta, através no item Advanced do menu Database, a geração dos dados testes é feita pela opção "Test data generator...". Em sua janela, este recurso permite a configuração dos valores que serão usados para cada coluna da tabela que receberá a carga de dados, bem como a quantidade de registros que serão produzidos. Qualquer configuração que venha a ser feita nesta tela poderá ser salva (no formato XML) e aproveitada em outras ocasiões. Ainda sobre a carga de dados, ela pode ser feita sobre uma ou várias tabelas.

Conclusão

No mundo do desenvolvimento em geral, e neste caso explicitando especialmente o do Delphi, nem sempre se faz presente no ambiente de trabalho o profissional desenvolvedor e o profissional administrador de banco de dados (DBA). É de praxe a empresa alocar somente o primeiro tipo de profissional e assim atribuir a ele também a função do segundo. Assim, cabe ao desenvolvedor saber dosar suas atribuições, responsabilidades com o seu tempo disponível e, dessa forma, buscar ferramentas que o ajudem neste objetivo. Neste ponto, o FlameRobin entra em ação, dispondo de toda sua característica simples e eficiente, conforme pôde ser vista neste artigo.

É sabido que existem outras inúmeras opções de ferramentas com estes mesmos propósitos, no entanto, cabe a você fazer o ideal discernimento sobre qual a ideal para sua atividade e seu ambiente. Mais importante ainda é o conhecimento de mais uma opção para o seu trabalho. Portanto fica a dica deste artigo.

Fabrício Hissao Kawata
fabricio.kawata@bol.com.br
Formado em Processamento de Dados pela FATEC-TQ. Atua como Analista Programador Delphi há 5 anos. Atualmente escreve artigos sobre Delphi para revistas especializadas além de participar do programa Beta Tester Delphi Prism, da Embarcadero.

FlameRobin - Página oficial
http://www.flamerobin.org

Firebird - Página oficial
http://www.firebirdsql.org

Firebase – Maior portal sobre Firebird no Brasil
http://www.firebase.com.br/fb

The Firebird FAQ
http://www.firebirdfaq.org

Firebird - Sourceforge
http://sourceforge.net/projects/firebird/files

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Personalizando aplicações WPF no Delphi Prism

Fabrício Hissao Kawata

O Delphi Prism é o atual IDE da Embarcadero para o desenvolvimento .NET, suportando as versões mais recentes do framework e utilizando o shell do Visual Studio como sua base principal. Dessa forma, tecnologias nativas Microsoft, como Windows Presentation Foundation (WPF), Silverlight, WCF e LINQ agora estão disponíveis também para o mundo Delphi. Tudo isso de forma facilitada, uma vez que a linguagem utilizada pelo Delphi Prism, a Oxygene (também tratada como D#), é toda baseada em Object Pascal, que é a linguagem do Delphi tradicional. Sendo assim, a curva de aprendizado é atenuada, se comparada a uma eventual mudança para o C#, por exemplo.

Nota do DevMan

Oxygene, anteriormente conhecida como Chrome, é uma linguagem de programação baseada em Object Pascal desenvolvida pela empresa RemObjects Software, voltada para o desenvolvimento .NET. Em meados de 2008, a RemObjects licenciou seus produtos (compilador e IDE) para a Embarcadero, resultando na IDE Delphi Prism, que se integra devidamente ao Visual Studio, aproveitando todos os recursos oferecidos pelo framework da Microsoft.

WPF

Com a reabertura do universo .NET para o Delphi, proporcionada pelo Delphi Prism, surgiu a possibilidade de construção de novos tipos de aplicações, no qual podemos citar as aplicações com interfaces ricas. Neste ponto surge o Windows Presentation Foundation, ou simplesmente WPF, que é o framework da Microsoft para a construção de aplicações que proporcionam um alto grau de experiência com o usuário, através de poderosos recursos visuais (incluindo gráficos 2D e 3D e animações) e suporte a multimídia (áudio e vídeo). Toda a definição visual do WPF se dá por meio de uma linguagem de marcação especial denominada XAML (eXtensible Application Markup Language), que é baseada em outra linguagem de marcação, a XML (Extensible Markup Language). Somam-se a isso, algumas outras vantagens oferecidas pela tecnologia que justificam sua adoção:

• Ao estudar o framework WPF, obtém-se um nível de conhecimento necessário não só para o desenvolvimento de aplicações, como também para o manuseio de segmentos tratados em separado por outras tecnologias, como é caso dos recursos de áudio, vídeo, animações, gráficos 2D e 3D;

• O WPF utiliza-se do processamento gráfico do computador, ao invés do processamento central, obtendo assim um melhor índice de renderização, resultando em características gráficas mais suaves e de mais qualidade;

• Os arquivos XAML, que definem toda sua interface visual, possuem um arquivo de code-behind associado, fazendo com que o código da interface de usuário fique separado do código de

Resumo DevMan

De que trata o artigo:

O artigo fará uma explanação de alguns dos recursos relacionados especificamente ao visual das aplicações WPF, através de quatro tópicos fundamentais: Styles, Templates, Skins e Themes. Estes recursos que até então eram figuras conhecidas no mundo do desenvolvimento Web como, por exemplo, em ASP.NET, agora estão disponíveis também para as aplicações Desktop por meio do WPF (Windows Presentation Foundation).

Em que situação o tema é útil:

Com o decorrer do tempo, durante o desenvolvimento de uma aplicação WPF, algumas necessidades em termos visuais são sentidas. Entre elas, podemos citar a definição de um padrão de estilização para os controles visuais, algum modelo referente à forma de como determinado tipo de controle deve ser apresentado na interface de usuário, bem como a capacidade da aplicação suportar a utilização de skins que alteram sua aparência, a critério do usuário final e de forma dinâmica. Todas essas necessidades são devidamente supridas por recursos oferecidos pelo próprio Windows Presentation Foundation e que serão abordados neste artigo através de Styles, Templates, Skins e Themes.

Styles, Templates, Skins e Themes no WPF:

A ideia do artigo é proporcionar ao leitor mais uma abordagem técnica de um, entre os inúmeros temas ligados à tecnologia WPF e à construção de aplicações deste tipo utilizando o Delphi Prism. O assunto escolhido, tem como foco principal, além da parte técnica e conceitual, mostrar as novas possibilidades que o casamento entre Delphi Prism e o .NET Framework pode proporcionar à comunidade Delphi. Esta união se torna ainda mais perfeita pelo fato do Delphi Prism ter o Oxygene como sua linguagem padrão, baseada no Object Pascal do Delphi. Além disso, por si só, o WPF inovou, disponibilizando recursos para o manuseio visual de aplicações Desktop, tais como Styles, Templates, Skins e Themes, que são itens que até então não eram tão comuns para este segmento. Para a demonstração de como cada um dos recursos pode ser empregado no WPF, será utilizada uma abordagem prática, através do uso de exemplos de situações fictícias.

negócio. Dessa forma, o trabalho entre os profissionais de designer e programação pode ser facilmente dividido;
• A composição visual oferecida pelo framework Windows Presentation Foundation proporciona ao desenvolvedor um nível de flexibilidade muito alta possibilitando, por exemplo, a composição de um controle através da combinação de outros (por exemplo, um Button contendo em seu interior um StackPanel e um TextBlock). Assim sendo, produzem-se controles customizados e únicos.

Ainda sobre XAML, na **Listagem 1** pode ser conferida a estrutura padrão de uma janela WPF, cuja definição se dá por meio desta linguagem de marcação.

Listagem 1. Código XAML que define uma janela padrão no WPF

```
<?xml version='1.0' encoding='utf-8' ?>
<Window x:Class="WPFApplication1.Window1"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Title="Window1" Height="300" Width="300">
  <Grid>
  </Grid>
</Window>
```

Outra observação a se fazer é que as aplicações WPF podem ser destinadas tanto para o ambiente Web, quanto Desktop, sendo esta última a mais empregada. Tanto é que a tecnologia é taxada como a sucessora das aplicações Windows Forms ("carro-chefe" da Microsoft, para aplicações Win32 até então). Além disso, alguns recursos que em outras tecnologias se aplicam exclusivamente a um ambiente ou outro, com o WPF não existe esta distinção (Web e Desktop). Neste contexto se encaixa a parte de padronização e estilização visual, oferecida por quatro recursos especiais denominados Styles, Templates, Skins e Themes, que podem ser popularmente traduzidos como estilos, modelos, peles e temas, respectivamente.

Styles

No desenvolvimento Web, o termo Style é relacionado à linguagem de estilo denominada CSS (Cascading Style Sheet). Ela é utilizada para definir a apresentação de documentos que utilizam determinada linguagem de marcação, como o HTML, que é o exemplo mais popular. Em termos técnicos, o CSS provê uma ideal separação entre o conteúdo e o formato do documento. O desenvolvedor pode, então, definir em um arquivo separado os estilos que serão usados por cada tipo de elemento presente no documento e depois apenas vincular este arquivo aos documentos HTML que deverão receber a estilização.

A estrutura de um estilo CSS, essencialmente, define as propriedades de um elemento e seus respectivos valores, formando assim um conjunto de regras. A seguir é mostrada uma declaração de um estilo simples para o elemento HTML body, onde são definidas sua margem e cor de fundo:

```
body {
    margin: 5px 10px;
    background-color: #0f0;
}
```

De forma semelhante ao que acontece com CSS e HTML, no WPF o principal objetivo de um estilo é a separação dos valores das propriedades dos elementos da interface de usuário, além de ser o ponto de partida, tecnicamente falando, para a definição dos recursos de Templates, Skins e Themes.

Templates

Novamente citando o desenvolvimento Web, os Templates são modelos de design pré-definidos que definem a apresentação visual de uma página Web, estipulando a separação entre layout e conteúdo. São também conhecidos como Web Templates. Em um template é possível, por exemplo, definir as imagens de decoração e toda a estrutura de linhas e colunas que uma página irá ter, bem como os locais onde o conteúdo desta deverá ser mostrado.

No caso do WPF, mais especificamente no caso deste artigo, serão abordados os templates relativos aos controles de uma aplicação, denominados Control Templates. Estes, assim como os Web Templates, têm a função de estabelecer um modelo para os elementos da interface visual. Através de Control Templates, o WPF permite, entre outras coisas, alterar a aparência visual por completo de determinado controle, sem que suas propriedades nativas sejam afetadas.

Um caso básico que pode ser citado é fazer com que um botão da aplicação seja exibido no formato redondo, em substituição à sua forma tradicional retangular. Para isso, é definido um Control Template que contém um elemento do tipo elipse, que por sua vez dará a forma desejada ao controle.

Skins e Themes

Em aplicações ASP.NET, os conceitos de Skins e Themes caminham juntos. O primeiro pode ser definido como um arquivo onde são definidas as customizações das propriedades dos controles visuais ASP.NET. Já o segundo padroniza e centraliza todas as regras de definição da aparência das páginas e controles da aplicação, sendo assim formado pelo conjunto de arquivos CSS, Skins, entre outros. Em suma, ambos proporcionam às aplicações a capacidade de mudança da aparência visual em tempo de execução num processo totalmente dinâmico.

No WPF, este dinamismo também é atingido, todavia, o contexto é um pouco diferente, uma vez que, Skins e Themes podem ser tratados de maneiras distintas, ao mesmo tempo em que se utilizam de técnicas semelhantes para serem implementados, conforme será visto mais adiante.

Utilizando estilos (Styles)

Styles (ou estilos) são entidades que agrupam valores de propriedades de determinado tipo de elemento, manuseando-as em conjunto de forma a partilhá-las com vários elementos ao mesmo tempo. Dessa forma são evitados ajustes individuais nas propriedades de cada elemento. No WPF, os estilos são representados pela classe System.Windows.Style.

Numa situação comum, onde se tem vários elementos do mesmo tipo numa aplicação, a maneira mais básica de personalizar suas propriedades é individualmente, conforme pode ser visto na **Listagem 2**.

Nota do DevMan

Windows Communication Foundation (WCF) é uma tecnologia que surgiu com a versão 3.0 do .NET Framework, fornecendo um modelo unificado de programação, voltado para a construção de aplicações distribuídas orientadas a serviço, também conhecidas como SOA.

Nota do DevMan

LINQ é a sigla para Language Integrated Query. Surgiu como parte integrante do .NET Framework 3.5, apresentando como característica principal permitir a realização de consultas a bancos de dados, vetores, coleções e outros, de modo integrado às linguagens C# e VB.NET. Sua sintaxe é inspirada na Structured Query Language (SQL), que é a linguagem padrão para comunicação com banco de dados relacionais.

Neste caso ilusório, suponha que seja necessário que todos os botões apresentados na interface visual de um aplicativo tenham a mesma aparência (**Figura 1**). Decisão esta que foi definida em reunião com o cliente: os botões devem ser coloridos (fora do tradicional), todos do mesmo tamanho e seu texto de exibição deve ser escrito com a fonte Verdana. Para este caso, utilizando a abordagem tradicional, somente por estes dois requisitos já se teria um enorme índice de retrabalho na aplicação, pois para cada botão que venha a ser utilizado, uma mesma porção de código deve ser replicada.

Figura 1. Aplicação com três botões personalizados

Para piorar, caso houver alguma mudança nos requisitos, a codificação de todos os controles do tipo Button deverá ser afetada, necessitando de intervenção manual em cada um.

Para a resolução de problemas como este, o WPF oferece o recurso de estilos. Então, de acordo com o exemplo apresentado, o que será feito é declarar as propriedades relacionadas aos requisitos exigidos para os elementos em um único lugar, definindo assim um estilo para o tipo Button. Consequentemente, apontaremos este Style a cada botão, simplificando o código da aplicação atual e facilitando posteriores alterações.

A **Listagem 3** mostra o trecho de código que é utilizado para a declaração de um novo estilo denominado "EstiloPadraoParaBotoes". Sua estrutura é composta de uma coleção de atributos Setter, que especificam o nome da propriedade de determinada classe e o valor desejado para ela. Por exemplo, a indicação Button.Background aponta para a propriedade Background da classe Button, cujo valor (Value) é a cor desejada que, neste caso, é Aquamarine.

Listagem 2. Trecho da codificação inicial da aplicação de exemplo WPF

```
<StackPanel Orientation="Horizontal" Width="270">
    <Button Name="button1" Content="Inserir"
        Background="Aquamarine" FontFamily="Verdana"
        Height="30" Width="90"
        HorizontalAlignment="Left" VerticalAlignment="Bottom" />
    <Button Name="button2" Content="Alterar"
        Background="Aquamarine" FontFamily="Verdana"
        Height="30" Width="90"
        HorizontalAlignment="Left" VerticalAlignment="Bottom" />
    <Button Name="button3" Content="Excluir"
        Background="Aquamarine" FontFamily="Verdana"
        Height="30" Width="90"
        HorizontalAlignment="Left" VerticalAlignment="Bottom" />
</StackPanel>
```

Listagem 3. Novo estilo para os botões

```
<StackPanel.Resources>
    <Style x:Key="EstiloPadraoParaBotoes">
      <Setter Property="Button.Background" Value="Aquamarine" />
      <Setter Property="Button.FontFamily" Value="Verdana" />
      <Setter Property="Button.Height" Value="30" />
      <Setter Property="Button.Width" Value="90" />
    </Style>
</StackPanel.Resources>
```

Uma vez definido o estilo, a **Listagem 4** mostra como é seu emprego no código do exemplo. Com a simples utilização de Styles na aplicação, o resultado desejado é atingido de forma mais simplificada, em termos de código, se comparado à abordagem tradicional. Assim, qualquer eventual mudança nos requisitos implicará na mudança somente nas propriedades do estilo. Por conseguinte, essas mudanças serão automaticamente refletidas nos elementos que fazem dele. Uma observação a se fazer é que caso uma declaração local de uma propriedade de um elemento coincida com outra presente no corpo do estilo, a primeira prevalece.

Ainda sobre estilos, o WPF proporciona o trabalho com a utilização de herança. Isso quer dizer que podemos definir um estilo que herde de outro. Este recurso é ideal quando necessitamos apenas incrementar um estilo existente, acrescentando uma propriedade específica, sem que isso afete os elementos que já o estejam usando.

Aproveitando esta ideia, supondo que em uma nova reunião com o cliente, definiu-se que especificamente para o novo botão Salvar da aplicação, além dos requisitos já definidos anteriormente, ele deve apresentar seu texto em negrito, para que ganhe um destaque necessário em tela. Obviamente, se esta mudança for feita diretamente sobre o estilo EstiloPadraoParaBotoes, todos os outros botões serão afetados, fugindo da necessidade real do cliente. Portanto, através do uso de herança, podemos definir um novo estilo, aproveitando a base do estilo existente, conforme mostrado na **Listagem 5**.

Pelo código apresentado, a chave para o uso de herança em estilos fica por conta da marcação BasedOn, onde é apontado o estilo base a ser considerado, fazendo com que todo o seu conjunto de Setters seja interpretado pelo novo estilo que está sendo criado.

Outra possibilidade ainda no WPF é compartilhar um mesmo estilo com outros tipos de elementos, ou seja, no caso do exemplo

proposto, o Style não ficaria restrito somente a elementos da classe Button. Supondo que nosso cliente fictício sugira que os elementos da aplicação como um todo tenham um padrão, caracterizado pela cor de fundo Aquarmarine e fonte Verdana, a solução se dá de forma fácil. Visto que o projeto já possui um estilo definido para isso, podemos alterá-lo de forma a atender elementos heterogêneos, como na **Listagem 6**.

Listagem 4. Novo estilo sendo aplicado no código do exemplo

```
<Button Name="button1" Content="Inserir"
    Style="{StaticResource EstiloPadraoParaBotoes}"
    HorizontalAlignment="Left" VerticalAlignment="Bottom" />
<Button Name="button2" Content="Alterar"
    Style="{StaticResource EstiloPadraoParaBotoes}"
    HorizontalAlignment="Left" VerticalAlignment="Bottom" />
<Button Name="button3" Content="Excluir"
    Style="{StaticResource EstiloPadraoParaBotoes}"
    HorizontalAlignment="Left" VerticalAlignment="Bottom" />
```

Listagem 5. Novo estilo baseado em um já existente

```
<Style x:Key="EstiloPadraoBotaoSalvar" BasedOn="{StaticResource
  EstiloPadraoParaBotoes}">
  <Setter Property="Button.FontWeight" Value="Bold" />
</Style>
```

Listagem 6. Definindo um estilo genérico

```
<Style x:Key="EstiloPadraoParaBotoes">
  <Setter Property="Control.Background" Value="Aquamarine" />
  <Setter Property="Control.FontFamily" Value="Verdana" />
  <Setter Property="Control.Height" Value="30" />
  <Setter Property="Control.Width" Value="90" />
</Style>
```

A única mudança significativa foi a da indicação de Button para Control, fazendo com que o WPF aplique o estilo devidamente em qualquer controle e não somente no tipo Button. Na **Figura 2** são exibidos outros tipos elementos (neste caso Label, TextBox e ComboBox) estilizados no mesmo padrão dos botões mostrados anteriormente. Um detalhe importante é que caso o elemento que receba o estilo não possua alguma das propriedades definidas no conjunto de Setters, esta mesma propriedade é prontamente ignorada pelo WPF, evitando possíveis erros e tornando o recurso "inteligente".

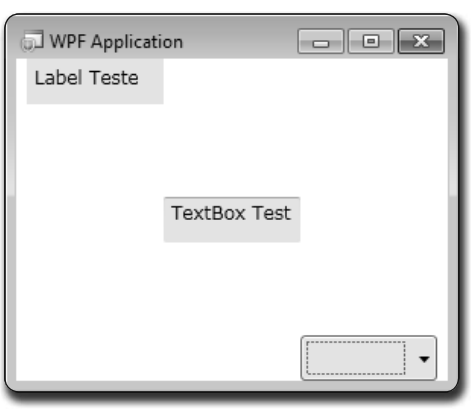


Figura 2. Outros elementos sendo estilizados

Da mesma forma que é possível "abrir" um estilo para tipos de elementos heterogêneos, pode-se também restringir seu uso a um tipo específico. Na **Listagem 7** é apresentada a declaração do mesmo estilo utilizado até aqui, agora restrito explicitamente para a classe Button (System.Windows.Controls.Button) e seus eventuais subtipos. Isto é definido pela tag TargetType. Como está sendo informado um tipo específico, no grupo de Setters não se faz mais necessária a indicação do nome da classe novamente, como foi feito anteriormente, mas somente o nome da propriedade. Uma vez definido isso, qualquer tentativa de atribuição que seja feita do estilo a outros tipos produzirá um erro em tempo de compilação, apresentado pelo IDE (**Figura 3**).

Figura 3. Erro ao tentar utilizar um estilo direcionado a um tipo específico em outro

Listagem 7. Definindo um estilo restrito

```
<Style x:Key="EstiloPadraoParaBotoes" TargetType="{x:Type Button}">
    <Setter Property="Background" Value="Aquamarine" />
    <Setter Property="FontFamily" Value="Verdana" />
    <Setter Property="Height" Value="30" />
    <Setter Property="Width" Value="90" />
</Style>
```

Estilos implícitos

Até agora foram mostradas formas de uso explícitas para os objetos Style, que é a mais comum. No entanto, o WPF apresenta outra maneira de trabalho com estilos, que é seu uso através da declaração implícita, em que não é necessária sua atribuição, via código, para cada elemento do contexto, conforme foi feito até aqui. Usando estilos implícitos, basta sua definição para que todo um escopo receba suas configurações. Utilizando nosso exemplo, imagine que seja necessário aplicar a estilização definida para os botões em toda a aplicação, ou seja, em todas as telas que o projeto venha a ter. No estado atual, seria necessária uma intervenção manual em todas as declarações de botão na aplicação para referenciar o estilo desejado. Uma solução é declarar o estilo na seção de Resources do objeto Application, visto na **Listagem 8**, presente no arquivo App.xaml, que representa o contexto da Aplicação em si.

Uma vez feito isso, de agora em diante todos os elementos do tipo Button do projeto já terão os estilos sendo aplicados de forma implícita, não sendo necessária qualquer declaração local.

Listagem 8. Definindo um estilo como um recurso da aplicação

```
<Application.Resources>
    <Style TargetType="{x:Type Button}">
      <Setter Property="Background" Value="Aquamarine" />
      <Setter Property="FontFamily" Value="Verdana" />
      <Setter Property="Height" Value="30" />
      <Setter Property="Width" Value="90" />
    </Style>
</Application.Resources>
```

Control Templates

De um modo geral, cada controle WPF possui uma gama de propriedades que permitem sua personalização de diferentes formas, compondo assim sua árvore visual. Basicamente, se você quer deixar um controle TextBox com o fundo na cor vermelha, basta declarar sua propriedade Background com o valor Red. No entanto, o WPF oferece um tipo de recurso especial para controles denominado Control Templates (modelos de controle), que possibilita a substituição por completo de sua árvore visual, podendo ser anexado então qualquer outro elemento WPF. Como exemplo, é possível definir uma árvore de elementos para compor a aparência de outro elemento. Tomando como base o nosso exemplo, suponha que o cliente solicitou uma nova mudança no visual da aplicação, de forma que os botões anteriormente definidos tenham uma nova aparência, agora arredondados e utilizando cores gradientes.

Nativamente, os botões apresentam uma forma retangular e cores sólidas, todavia, podemos criar um Control Template para a resolução do problema. Uma codificação sugerida pode ser vista na **Listagem 9**. Nela, fica nítido o envolvimento de outros elementos (Grid e Ellipse) para sua composição.

Listagem 9. Definindo um Control Template

```
<ControlTemplate x:Key="ModeloPadraoParaBotoes">
<Grid>
  <Ellipse Height="30" Width="90">
        <Ellipse.Fill>
          <LinearGradientBrush StartPoint="0,0" EndPoint="0,2">
            <GradientStop Offset="0" Color="Aquamarine"/>
            <GradientStop Offset="2" Color="Blue"/>
          </LinearGradientBrush>
        </Ellipse.Fill>
      </Ellipse>
    </Grid>
</ControlTemplate>
```

Sua aplicação nos controles Button se dá de maneira semelhante ao uso de estilos, mas dessa vez utiliza-se a marcação Template, conforme é exibido no trecho a seguir. A **Figura 4** mostra o resultado desta operação:

```
<Button Name="button1" Content="Inserir"
    Template="{StaticResource ModeloPadraoParaBotoes}"
    HorizontalAlignment="Left" VerticalAlignment="Bottom" />
```

Figura 4. Aplicação com botões arredondados

Apesar de o objetivo ter sido atingido, com os botões assumindo a nova forma desejada, percebe-se pelo seu resultado visual que algo ficou faltando. O texto dos botões foi simplesmente ocultado após o emprego do Control Template e isso não é desejável para o projeto, uma vez que, este modelo será empregado em todos os demais botões da aplicação. Uma solução elegante para este tipo de adversidade é aproveitar os recursos de Data Binding (ligação de dados) do WPF, mais especificamente através da classe TemplateBindingExtension. No código, ela é declarada simplesmente como TemplateBinding e é customizada para o uso de templates. Em termos estruturais, um TemplateBinding tem como fonte de dados (Data Source) o próprio elemento destino, e pode ser direcionado a qualquer propriedade envolvida ao elemento. Aproveitando o nosso próprio exemplo, o que se deseja é mostrar o conteúdo do texto do botão dentro do próprio botão, dessa forma, agora com o uso de TemplateBinding o código do Control Template definido anteriormente ficaria como o apresentado na **Listagem 10**.

Listagem 10. Control Template utilizando Template Binding

```
<ControlTemplate x:Key="ModeloPadraoParaBotoes">
 <Grid>
   <Ellipse Height="30" Width="90">
        <Ellipse.Fill>
          <LinearGradientBrush StartPoint="0,0" EndPoint="0,2">
            <GradientStop Offset="0" Color="Aquamarine"/>
            <GradientStop Offset="2" Color="Blue"/>
          </LinearGradientBrush>
        </Ellipse.Fill>
   </Ellipse>
    <Viewbox>
     <ContentControl Margin="8" Content="{TemplateBinding Button.Content}"/>
    </Viewbox>
 </Grid>
</ControlTemplate>
```

Conforme o novo código apresentado, foi utilizado um novo elemento Viewbox que possui uma propriedade aninhada ContentControl, cuja tag Content exibe o texto do botão em tela, utilizando para isso a marcação de Template Binding.

Para não deixar de mencionar, da mesma forma que os estilos, um Control Template pode ser restringido a um tipo específico de controle, através de sua marcação TargetType. Caso fosse estabelecido que o template construído devesse ser direcionado somente a elementos do tipo Button, sua declaração deveria ser feita da seguinte forma:

```
<ControlTemplate x:Key="ModeloPadraoParaBotoes" TargetType="{x:Type Button}">
...
</ControlTemplate>
```

Um fato a ser levado em consideração com relação à criação de Control Templates é que, intuitivamente, o objetivo de se criar "modelo" é que ele seja o mais reutilizável possível. No entanto, numa aplicação, por mais simples que seja sempre haverá uma enorme gama de controles sendo utilizados e atingir uma padronização ideal sem que haja exceções não é uma tarefa fácil.

Dessa forma, não importa o tipo de controle ao qual o Control Template está destinado, inevitavelmente haverá a necessidade

de que certas propriedades locais de algum controle sejam preservadas. Pondo isto na prática, suponha que o Control Template criado para os botões, ao invés de definir, de forma explícita a cor inicial do efeito gradiente, fizesse uso da própria cor do controle, definida em suas propriedades locais.

Assim sendo, teríamos os botões da aplicação todos com o mesmo efeito de preenchimento de cor (gradiente), cuja cor inicial é a definida nas propriedades do próprio controle e não no modelo. Na **Listagem 11** é apresentado um trecho de código que faz uso deste recurso. A marcação GradientStop é quem irá receber o valor, em seu atributo Color, da propriedade do elemento alvo ao qual o template está sendo aplicado. Para que o Control Template saiba exatamente em qual propriedade buscar esse valor, é realizada uma ligação (Binding) com a propriedade Background (marcação Path) do elemento alvo (TemplatedParent). Todo esse processo pode ser visto nas linhas 07 e 08.

Listagem 11. Control Template que faz uso de propriedade do controle destino

```
01 <ControlTemplate x:Key="ModeloPadraoParaBotoes" TargetType=
     "{x:Type Button}">
02   <Grid>
03     <Ellipse>
04       <Ellipse.Fill>
05         <LinearGradientBrush StartPoint="0,0" EndPoint="0,2">
06             <GradientStop Offset="0"
07               Color="{Binding RelativeSource=
08               {RelativeSource TemplatedParent}, Path=Background.Color}"/>
09             <GradientStop Offset="2" Color="Blue"/>
10         </LinearGradientBrush>
11       </Ellipse.Fill>
12     </Ellipse>
13   </Grid>
14 </ControlTemplate>
```

Nota

O leitor que possui algum conhecimento técnico sobre WPF, mais especificamente sobre Control Templates, já deve ter observado em alguns artigos técnicos na Web ou através de livros, que o elemento ContentControl, algumas das vezes é substituído por ContentPresenter, na estrutura de um Control Template. Isso pode ser justificado por ambas apresentarem o mesmo fim, que é a exibição de conteúdo. Entretanto, alguns autores recomendam o uso de ContentPresenter ao invés de ContentControl por se tratar de um elemento "mais leve" e primitivo, e projetado exclusivamente para Control Templates. No entanto, o que prevalece ainda é a preferência do próprio desenvolvedor.

Feito isso, exemplificando o uso deste Control Template sobre os três botões utilizados até aqui, teríamos uma resultado visual igual ao da **Figura 5**, onde para cada botão simplesmente foi definida uma cor de fundo (background) diferente, refletindo diretamente no gradiente realizado pelo template, obtendo-se três efeitos diferentes.

Triggers com Control Templates e Styles

De acordo com o exemplo, outro "problema" apresentado pelos botões com o uso de Control Templates é seu comportamento visual em runtime. Apesar de todas suas funcionalidades estarem OK, o seu "Click" fica ofuscado pelo template, fazendo-o se parecer como uma simples imagem em tela, tirando do usuário a sensação de estar efetivamente clicando num botão. Neste tipo de situação, o uso de Triggers com Control Templates faz com que esta sensação seja retomada.

Figura 5. Três gradientes diferentes a partir de um único Control Template

A estrutura de um Trigger (gatilho), assim como o estilo, é composta de um conjunto de marcadores Setters, que definem suas propriedades. Para que um gatilho seja disparado e realize suas tarefas é necessário que uma condição seja respeitada. Exemplificando, pode-se definir um Trigger que é disparado no passar do mouse por um controle TextBox, mudando a cor de fundo do controle. Neste caso, o passar do mouse é a condição e a mudança de cor sua tarefa.

Tomando como base a necessidade de nosso exemplo, é preciso que os botões voltem a ter, visualmente, o comportamento de pressionamento. Para a utilização de Triggers no Control Template existente, basta declarar um novo bloco de código em seu corpo, conforme apresentado na **Listagem 12**.

O uso de Triggers também pode ser feito com estilos, de maneira muito parecida com seu uso com Control Templates. Para ilustrar um caso prático, o código da **Listagem 13** apresenta a declaração de um estilo, cujo gatilho é disparado ao passar do mouse sobre um controle TextBox, alterando sua cor de fundo e negritando seu texto.

Mesclando estilos e modelos

Outra possibilidade oferecida pelo WPF é de ter objetos Styles que possuem em sua estrutura outros objetos do tipo Control Template. Uma utilização comum para esta "mistura" é poder definir um estilo personalizado para uma determinada propriedade, ao mesmo tempo em que esta é usada como padrão para a definição de um modelo para certo tipo de elemento. Vendo isto na prática, a **Listagem 14** define um novo objeto Style exclusivo para o tipo Label, onde sua propriedade BorderBrush é estilizada com uma cor padrão simples. Na mesma estrutura, é definido um Control Template, que transforma o Label em forma de elipse, usando como sua base de preenchimento a cor definida em BorderBrush anteriormente.

Ao utilizar esta combinação de recursos, outro tipo de customização é alcançado. Analisando o comportamento do resultado da codificação apresentada, os controles Label que fizerem uso do estilo definido terão o formato arredondado (de elipse) e sem borda, visto que a cor desta é a mesma usada em seu preenchimento.

Ao mesmo tempo, se algum controle Label possuir a propriedade BorderBrush definida localmente, esta será a cor usada também em seu preenchimento, devido à indicação de TemplateBinding no Control Template. Na **Figura 6** são apresentados três elementos Label que fazem uso do estilo da **Listagem 14**, no entanto, Label1 possui uma cor diferente, pois sua propriedade BorderBrush é definida localmente, como visto a seguir:

```
<Label Style="{StaticResource EstiloTextBox}" Content="Label1" BorderBrush="Yellow" />
<Label Style="{StaticResource EstiloTextBox}" Content="Label2" />
<Label Style="{StaticResource EstiloTextBox}" Content="Label3" />
```

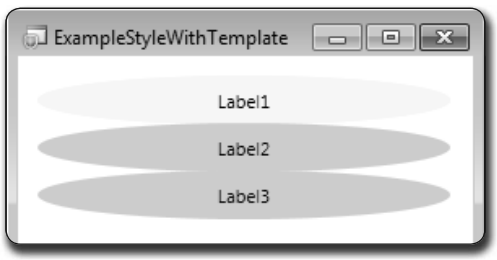


Figura 6. Exemplo de uso da mescla de Style e Control Template

Listagem 12. Utilizando Trigger com Control Template

```
<ControlTemplate.Triggers>
  <Trigger Property="Button.IsPressed" Value="True">
    <Setter Property="RenderTransform">
      <Setter.Value>
        <ScaleTransform ScaleX=".8" ScaleY=".8"/>
      </Setter.Value>
    </Setter>
  <Setter Property="RenderTransformOrigin" Value=".6,.6"/>
  </Trigger>
</ControlTemplate.Triggers>
```

Listagem 13. Utilizando Trigger com Style

```
<Style x:Key="EstiloComTriggerParaTextBox" TargetType="{x:Type TextBox}">
  <Style.Triggers>
    <Trigger Property="IsMouseOver" Value="True">
      <Setter Property="Background" Value="Yellow" />
      <Setter Property="FontWeight" Value="Bold" />
    </Trigger>
  </Style.Triggers>
</Style>
```

Listagem 14. Style com Control Template

```
<Style x:Key="EstiloTextBox" TargetType="{x:Type Label}">
  <Setter Property="BorderBrush" Value="LightGray"/>
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="{x:Type Label}">
          <Grid>
          <Ellipse Fill="{TemplateBinding BorderBrush}" />
          <Viewbox>
              <ContentControl Margin="8"
              Content="{TemplateBinding Content}"/>
          </Viewbox>
          </Grid>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

Utilizando Skins

O WPF, apesar de não possuir um mecanismo exclusivo para o trabalho com Skins, o faz de maneira satisfatória através de alguns de seus recursos dinâmicos, em combinação com os recursos de Styles e Templates apresentados. Dessa forma, uma aplicação fica apta a trocar de aparência a qualquer momento, abstraindo inclusive toda sua configuração visual em um arquivo separado, tornando-o utilizável também em outros projetos.

Em termos práticos, para implementar um mecanismo de Skin numa aplicação WPF, são necessários basicamente quatro etapas. Antes de falar sobre cada uma, vamos tomar como base uma aplicação WPF cujo visual se pareça com o da **Figura 7**. Dito isto, a primeira etapa é definir de forma global os estilos que serão usados para cada tipo de controle presente na aplicação. Como já visto anteriormente, isso é possível ao fazer a declaração dos elementos Style na área de Resources do objeto Application, presente no arquivo App.xaml. Neste momento, como exemplo, vamos definir que todos os tipos de elementos utilizados na aplicação serão estilizados: Grid, Label, TextBox e Button. A **Listagem 15** apresenta um esboço dos estilos para cada tipo de elemento sendo definidos como recursos da aplicação.

Figura 7. Visual da aplicação de exemplo

Todos estes estilos definidos estabelecem um padrão visual inicial para a aplicação. A seguir a segunda etapa consiste em fazer com que os elementos da aplicação façam uso desta estilização definida. A **Listagem 16** apresenta o código do exemplo já adaptado para esta situação, fazendo uso de estilos, conforme já apresentado anteriormente, porém com um detalhe importante: nas marcações de Style de cada elemento, ao invés de StaticResource, é utilizado DynamicResource que, como seu próprio nome sugere, aponta para um recurso dinâmico. Esta é a marcação que deve ser utilizada para que um estilo possa ser alterado dinamicamente.

Uma vez que os elementos já estão prontos para utilizar recursos dinâmicos, a terceira etapa se refere à criação do arquivo de um dos vários possíveis arquivos de Skin que uma aplicação poderá trabalhar. Cada arquivo deste nada mais é que um arquivo XAML comum contendo código de marcação específico para o seu fim, representado pela tag ResourceDictionary. Novamente, a nomenclatura associada é bem intuitiva, e pode ser interpretada como um dicionário de recursos diversos. Para este caso em específico,

Listagem 15. Código que define os estilos a serem usados por toda a aplicação

```
<Application.Resources>
    <Style x:Key="EstiloParaGrid" TargetType="{x:Type Grid}">
      <Setter Property="Background" Value="LightBlue"/>
    </Style>
    <Style x:Key="EstiloParaLabel" TargetType="{x:Type Label}">
      <Setter Property="FontWeight" Value="Bold"/>
      <Setter Property="Background" Value="Aqua"/>
    </Style>
    <Style x:Key="EstiloParaTextBox" TargetType="{x:Type TextBox}">
      <Setter Property="FontStyle" Value="Italic"/>
      <Setter Property="Background" Value="Aquamarine"/>
    </Style>
    <Style x:Key="EstiloParaButton" TargetType="{x:Type Button}">
      <Setter Property="FontWeight" Value="Bold"/>
      <Setter Property="Background" Value="Aquamarine"/>
    </Style>
</Application.Resources>
```

Listagem 16. Código da janela principal da aplicação de exemplo

```
<Grid Style="{DynamicResource EstiloParaGrid}">
    <TextBox Name="textBox1"
         Height="23" Width="75"
         HorizontalAlignment="Left" VerticalAlignment="Top"
         Margin="68,67,0,0"
         Style="{DynamicResource EstiloParaTextBox}"/>
    <TextBox Name="textBox2"
         Height="23" Width="165"
         HorizontalAlignment="Left" VerticalAlignment="Top"
         Margin="68,96,0,0"
         Style="{DynamicResource EstiloParaTextBox}"/>
    <Label Name="label1" Content="Código"
        Height="28"
        HorizontalAlignment="Left" VerticalAlignment="Top"
        Margin="12,65,0,0"
        Style="{DynamicResource EstiloParaLabel}"/>
    <Label Name="label2" Content="Nome"
        Height="28"
        HorizontalAlignment="Left" VerticalAlignment="Top"
        Margin="18,96,0,0"
        Style="{DynamicResource EstiloParaLabel}"/>
    <Button Name="button1" Content="Salvar"
         Height="23" Width="75"
         HorizontalAlignment="Left" VerticalAlignment="Top"
         Margin="59,165,0,0"
         Style="{DynamicResource EstiloParaButton}"/>
    <Button Name="button2" Content="Cancelar"
         Height="23" Width="75"
         HorizontalAlignment="Left" VerticalAlignment="Top"
         Margin="149,165,0,0"
         Style="{DynamicResource EstiloParaButton}"/>
  </Grid>
```

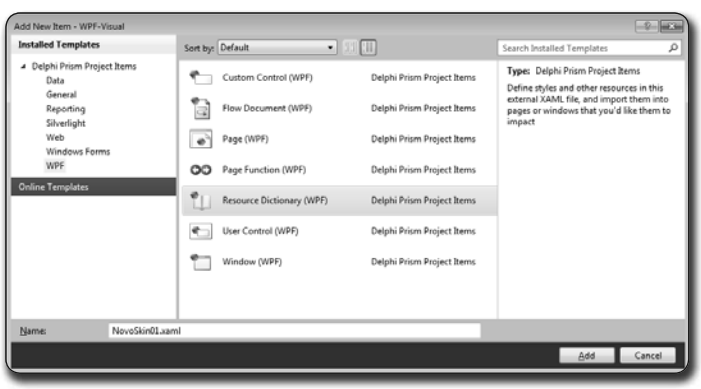


Figura 8. Criando um arquivo Resource Dictionary pelo IDE

para recursos de estilização visual. Arquivos deste tipo, aliás, já possuem um modelo pré-pronto oferecido pelo próprio IDE (**Figura 8**). Na **Listagem 17**, é exemplificada a codificação de um novo arquivo de "skin" para a aplicação de exemplo.

Listagem 17. Codificação do arquivo de "Skin"

```
<?xml version='1.0' encoding='utf-8' ?>
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
  presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   >
   <Style x:Key="EstiloParaGrid" TargetType="{x:Type Grid}">
     <Setter Property="Background" Value="YellowGreen"/>
   </Style>
   <Style x:Key="EstiloParaLabel" TargetType="{x:Type Label}">
     <Setter Property="FontWeight" Value="Bold"/>
     <Setter Property="Background" Value="Yellow"/>
   </Style>
   <Style x:Key="EstiloParaTextBox" TargetType="{x:Type TextBox}">
     <Setter Property="FontStyle" Value="Italic"/>
     <Setter Property="Background" Value="LightYellow"/>
   </Style>
   <Style x:Key="EstiloParaButton" TargetType="{x:Type Button}">
     <Setter Property="FontWeight" Value="Bold"/>
     <Setter Property="Background" Value="LightYellow"/>
   </Style>
</ResourceDictionary>
```

Conforme pode ser observado até este momento, em termos de definição estrutural a implementação do recurso de Skin na aplicação já está pronta. O que resta agora é apenas realizar a codificação necessária para que a aplicação possa interpretar o arquivo de Skin criado. Em aplicações reais, é comum ver este tipo de recurso sendo disponibilizado em janelas ou menus de configuração, uma vez que Skins são voltados principalmente ao usuário final.

Contudo, numa aplicação teste, como neste caso, toda a codificação pode ser colocada simplesmente no evento Click de um botão, como mostrado na **Listagem 18**. Todo código envolvido é bem simplificado, cabendo fazer uma observação à linha 07, onde o arquivo de Skin criado é recuperado em uma variável local de Stream. Já na linha 09, é feito um cast do conteúdo do arquivo XAML de Skin, para um objeto da classe System.Windows.ResourceDictionary, que é usado na linha 10 para atualizar a coleção de recursos da aplicação, definida anteriormente na **Listagem 15**.

Listagem 18. Codificação do botão para mudar o Skin da aplicação

```
01 method ExampleSkins.btnMudarSkin_Click(sender: System.Object;
02  e: System.Windows.RoutedEventArgs);
03 var
04  recursos: ResourceDictionary;
05  arquivoStream: FileStream; //dar uses em System.IO
06 begin
07    arquivoStream := new FileStream('NovoSkin01.xaml', FileMode.Open,
      FileAccess.Read);
08    //dar uses em System.Windows.Markup
09    recursos := (XamlReader.Load(arquivoStream) as ResourceDictionary);
10    Application.Current.Resources := recursos;
11 end;
```

Na verdade, toda a declaração presente em Application.Resources é descartada, sendo assim, caso você queira que a aplicação possa voltar para o seu visual padrão dinamicamente, deverá armazenar o dicionário anterior de alguma forma. Feito isso, o mecanismo de Skin está devidamente implementado. Tanto é que, com base no exemplo proposto, ao click do botão Mudar Skin, pode ser conferida a mudança do visual da aplicação em tempo de execução.

Utilizando Temas

Temas (Themes) é outro recurso apresentado pelo WPF para customização das interfaces visuais do usuário. Um Theme tem como peculiaridade o poder de refletir nos elementos da aplicação, características visuais do Sistema Operacional. Dessa forma, pode-se definir que enquanto a personalização com uso Skins afeta somente aplicações específicas, que fazem uso explícito deles, os Themes se refletem em qualquer que seja a aplicação. A simples mudança do tema do Windows afeta diretamente algumas características dos controles WPF, como cor e brilho. Dito isto, nota-se a importância em ter conhecimento sobre o assunto e assim atingir um nível ideal de coerência entre seus controles personalizados com o padrão de tema utilizado pelo Sistema Operacional.

Tomando como base a construção de um estilo, que é a forma mais simplista de se personalizar um determinado grupo de elementos, suponha a seguinte proposta: o estilo deverá proporcionar uma cor gradiente de fundo exclusivamente para controles StackPanel. Uma vez tendo o conhecimento sobre Themes, podemos definir este novo estilo de forma a aproveitar adequadamente as cores do tema corrente do Windows. Na **Listagem 19** temos o código de um estilo que faz proveito de tais recursos.

Listagem 19. Aproveitando as cores definidas no tema do Sistema Operacional

```
01 <Style x:Key="EstiloStackPanel" TargetType="{x:Type StackPanel}">
02   <Style.Resources>
03     <RadialGradientBrush x:Key="EfeitoGradiente">
04       <GradientStop Offset="0"
05       Color="{DynamicResource {x:Static SystemColors.InactiveCaptionColorKey}}"/>
06       <GradientStop Offset="1"
07       Color="{DynamicResource {x:Static SystemColors.ActiveCaptionColorKey}}"/>
08     </RadialGradientBrush>
09   </Style.Resources>
10
11   <Setter Property="Background" Value="{StaticResource EfeitoGradiente}" />
12 </Style>
```

Em vista do que foi apresentado, vale fazer alguns comentários, principalmente ao código das linhas 05 e 07. Para a definição do gradiente do estilo, são estipuladas as duas cores base para o efeito que, neste caso são cores do tema corrente do sistema. Assim, o valor da propriedade Color de cada elemento GradientStop, é marcada como um recurso dinâmico (DynamicResource), apontado para a cor do sistema através de SystemColors. Esta marcação recupera uma cor específica, dentre as várias definidas no contexto do Sistema Operacional, como por exemplo, InactiveCaptionColorKey e ActiveCaptionColorKey.

O resultado disto pode ser conferido a seguir, onde é retratada a janela de uma aplicação contendo quatro elementos StackPanel que fazem uso do estilo criado. São duas imagens da mesma aplicação, sendo que a primeira (**Figura 9**) mostra a aplicação sendo executada em um sistema Windows que utiliza o tema clássico (Classic). Já a segunda (**Figura 10**) imagem roda em um sistema cujo tema é o Windows 7 Basic.

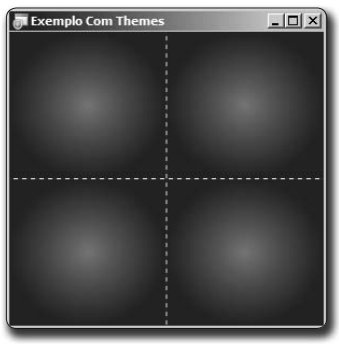


Figura 9. Aplicação rodando no tema Classic

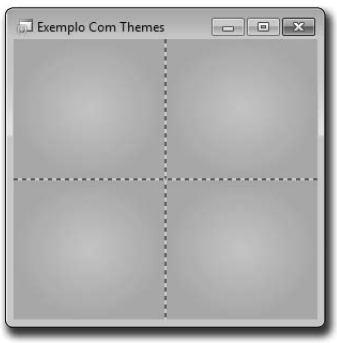


Figura 10. Aplicação rodando no tema Windows 7 Basic

Veja que toda a mudança de visual realizada acontece de maneira natural, sem intervenção programática alguma. Isso se deve pelo fato do WPF possuir e utilizar dicionários de temas internos, realizando a troca automática quando ocorre a mudança do tema do Sistema Operacional. Devido a este processo, abre-se mais um precedente, que são os casos onde o desenvolvedor queira ter controle total sobre o tema a ser aplicado. Em outras palavras, ele deve ser capaz de definir arquivos de temas (semelhantes aos de Skin) e poder estipular qual será aplicado em cada opção de tema do Windows.

O WPF oferece uma maneira eficaz para que isso seja feito. Num primeiro momento pode até se imaginar que exista algum evento no contexto do objeto Application, exclusivo para este tipo de ação como, por exemplo, um OnThemeChanging. Todavia, o WPF utiliza uma abordagem mais simples. Na prática, isso envolve alguns passos essenciais.

Utilizando uma forma básica, o ponto de partida é a definição dos arquivos XAML de ResourceDictionary (dicionário de recursos), sendo um arquivo separado para cada tema do Windows que se queira substituir. Nessa ocasião, este tipo de recurso é tratado com um dicionário temático. A segunda etapa é utilizar o esquema padrão de nomenclatura específico para estes arquivos: NomeTema.CorTema.xaml. Este esquema é utilizado para que o WPF consiga identificar o arquivo que será usado a cada tema do Sistema Operacional. Por padrão e de acordo com a versão do Windows, alguns dos possíveis nomes a serem utilizados são:

- Windows XP:

–Luna.NormalColor.xaml – tema padrão;
–Luna.Homestead.xaml – tema Olive Green;
–Luna.Metallic.xaml – tema Silver;
–Royale.NormalColor.xaml – tema Media Center Edition 2005;
–Classic.xaml – tema clássico;
–Zune.NormalColor.xaml – tema Zune;

• Windows 7 e Windows Vista:
–Aero.NormalColor.xaml – tema Aero;

Além destas opções, há ainda uma opção genérica, nomeada como "Generic.xaml" que deve ser criada para que o WPF a utilize quando determinado tema do Sistema Operacional não possua um arquivo de dicionário definido. Continuando, é necessário salvar estes arquivos XAML em uma subpasta que deve estar contida na raiz do projeto, nomeada como "themes". Uma vez definido o local onde estarão todos os dicionários de temas personalizados, é preciso preparar a aplicação para que ela tome conhecimento sobre esta localização, de forma a automatizar o mecanismo de interpretação de cada tema. Essa definição é feita em nível de assembly, que em um projeto WPF se dá pelo arquivo AssemblyInfo, através do uso do atributo ThemeInfo. O construtor deste atributo recebe dois parâmetros do tipo ResourceDictionaryLocation que justamente se referem à localização dos seus dicionários temáticos customizados. O primeiro parâmetro indica o local da coleção e o segundo a localização do dicionário de recursos genérico. A seguir, é mostrado um exemplo da declaração de ThemeInfo, que aponta a localização dos dicionários para a raiz do projeto, e que por padrão se refere à pasta "themes", citada anteriormente:

```
[assembly: ThemeInfo(ResourceDictionaryLocation.SourceAssembly, Resource
Dictionary
Location.SourceAssembly)]
```

Como visto, ResourceDictionaryLocation utilizou um tipo específico, denominado SourceAssembly, para fazer referência à raiz do assembly. Além deste, outros dois tipos são suportados:
• None – Este é o valor padrão e faz com que dicionários de recursos não sejam procurados;
• ExternalAssembly – Este tipo faz a procura dos dicionários em outros assemblies que, como regra, devem possuir o seguinte padrão de nomeação: NomeDoAssembly.NomeDoTema.dll. Este, aliás, é o tipo utilizado pela estrutura interna do WPF para a definição de seus dicionários temáticos. Só para citar, PresentationFramework.Luna.dll é o assembly utilizado pelo WPF, para o tema Aero do Windows 7.

Para terminar, a aplicação WPF também deve ser capaz de interpretar este novo conjunto de dicionários temáticos. Isto é feito na seção de recursos da aplicação (Application.Resources), onde para a fonte (Source) do dicionário (ResourceDictionary) é utilizada uma extensão de marcação (markup extension) chamada ThemeDictionary, que faz a referência para o conjunto de dicionários definidos internamente à aplicação. A **Listagem 20** mostra um exemplo simples desta codificação. Depois de todo este processo, a cada mudança de tema do Sistema Operacional, um novo dicionário temático relativo é utilizado pela aplicação.

Listagem 20. Definindo um novo conjunto de dicionários temáticos em Application.Resources

```
<Application.Resources>
    <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="{ThemeDictionary NomeDaAplicacao}"/>
      </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
</Application.Resources>
```

Conclusão

O Windows Presentation Foundation tem como um de seus principais atrativos o fator visual, graças a sua forma peculiar em renderizar seus elementos gráficos, que o diferem de qualquer outro tipo de aplicação. Como consequência, torna-se agradável ao usuário final ver uma aplicação WPF em execução, em vista das experiências visuais que lhe são proporcionadas. Mediante isso, a tecnologia oferece ainda grandes vantagens também aos profissionais envolvidos na construção deste tipo de aplicação, como é o caso dos desenvolvedores e designers. Algumas delas foram apresentadas neste artigo como, mais precisamente em quatro tópicos principais: Styles, Templates, Skins e Themes. No entanto, apesar de poderosos, a possibilidade destes recursos serem mesclados e combinados pode gerar confusão ao utilizá-los num primeiro momento. Apesar disso, com o devido conhecimento técnico, muitos benefícios são proporcionados, tais como permitir que o usuário final mude a aparência da aplicação dinamicamente ou a definição de uma estilização padrão por completo em um projeto. Ganha-se tanto em produção quanto na funcionalidade do aplicativo que está sendo desenvolvido.

Após a leitura do artigo, caso surja o interesse em se aprofundar mais nestas questões visuais do WPF, uma dica é fazer um estudo sobre a ferramenta Microsoft Expression Blend, voltada exclusivamente para criação de interfaces ricas para WPF e Silverlight. Ela, aliás, "conversa" diretamente com a IDE do Visual Studio (Delphi Prism), podendo manipular um mesmo projeto nas duas ferramentas.

Fabrício Hissao Kawata
fabricio.kawata@bol.com.br
Formado em Processamento de Dados pela FATEC-TQ. Atua como Analista Programador Delphi há 5 anos. Atualmente escreve artigos sobre Delphi para revistas especializadas além de participar do programa Beta Tester Delphi Prism, da Embarcadero.

Delphi Prism XE - Página de apresentação do produto
http://www.embarcadero.com/products/delphi-prism

Delphi Prism XE – Página para download
https://downloads.embarcadero.com/free/delphi_prism

Microsoft WPF - Página oficial
http://windowsclient.net

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Morfik, IDE para desenvolvimento Web usando Object Pascal

Carlos Eduardo Paulino

A capacidade de adaptar-se rapidamente é tudo hoje em dia, mas as ferramentas e tecnologias utilizadas para desenvolver aplicativos sofisticados baseados na web estão cada vez mais mostrando fraquezas com a crescente demanda de aplicativos e funcionalidades web. Um processo que já não é fácil nem rápido vem se tornando cada vez mais difícil.

Pensando nisso, Morfik está redefinindo-a partir do zero, como os aplicativos web são construídos. Morfik introduz em seu IDE tecnologias de outras áreas de especialização de TI, tais como o desenvolvimento de aplicativos tradicionais, design gráfico, animações gráficas e aplicativos de produtividade. Morfik é uma combinação de elementos de todas estas áreas para o desenvolvimento de aplicações web, a fim de tornar todo o processo mais produtivo.

O conceito de Morfik

Diferente do modo feito pelas ferramentas tradicionais, Morfik introduz uma nova forma de criar aplicações web de alto-nível, linguagens orientadas a objetos e ferramentas de design visual. Com este novo conjunto de ferramentas, os desenvolvedores podem construir suas aplicações utilizando as práticas de desenvolvimento de software utilizando o que há de melhor e seja de seu domínio. Podem aproveitar a sua experiência enquanto se beneficiam de um design visual moderno e produtivo. Isso faz com que a criação de web, interfaces interativas, sejam leves e muito fáceis.

Ao utilizar Morfik 3.x, os desenvolvedores podem ter a mesma produtividade como eles têm utilizando suas habilidades e conhecimentos já adquiridos com a prática de desenvolvimento de software desktop ou mesmo web. Os desenvolvedores podem usar sintaxes de linguagens familiares de alto nível, como Pascal, C# e Basic, no modo procedural ou orientado a objetos, então Morfik irá compilar, interpretando e gerando a aplicação final. Ao lado do navegador o aplicativo é compilado em uma combinação de HTML, XML, CSS e JavaScript, enquanto o código destinado a parte de controle de aplicativos (servidor) do aplicativo é compilado em um binário nativo executável, plugins para servidores IIS/Apache, ou bibliotecas DLLs. Para tornar as coisas ainda mais fáceis, o compilador Morfik também leva em conta as informações de design visual e gera as imagens necessárias para mostrar os efeitos desejados.

Suporte aos principais navegadores do mercado

O conteúdo HTML/XML/CSS combinado com código JavaScript gerado pelo Morfik para executar no navegador é inteiramente baseado em padrões de compatibilidades, a comunicação entre os componentes do servidor Morfik é feita através de solicitações HTTP nos padrões de serviço web.

As aplicações criadas em Morfik podem ser usadas para fazer interface com os diferentes serviços que estão disponíveis na Web ou simplesmente para dividir projetos complexos envolvendo diferentes equipes trabalhando na interface (camada de

Resumo DevMan

De que trata o artigo:

O artigo trata da apresentação do IDE Morfik, inspirada em Delphi, onde são mostrados os principais recursos de uma solução completa para o desenvolvimento de sistemas web. Uma única plataforma que permite integração com banco de dados, Web Services, construção de design, codificação de regras de negócio Client/Server e sua publicação, utilizando linguagem Object Pascal, C# ou Visual Basic.

Em que situação o tema é útil:

É útil para desenvolver sites tanto para navegadores desktop quanto para dispositivos móveis e principalmente que sejam compatíveis com os servidores mais utilizados como IIS e Apache em Windows ou Linux e ainda nas nuvens. Partindo do zero, reutilizando seus produtos desktop, criando integrações com sistemas de terceiros, consumindo e publicando serviços web.

Morfik IDE Web Object Pascal:

As aplicações web estão tomando cada vez mais o lugar de muitas soluções desktop. O profissional desktop muitas vezes tem dificuldade para entrar no mundo de aplicativos web, devido à falta de conhecimento em conceitos e linguagens web, que embora estejam cada vez mais poderosas, ainda assim não se comparam a produtividade do desenvolvimento desktop. A plataforma Morfik tem um conceito muito diferente, pois permite ao profissional desenvolver uma solução web sem se preocupar com estas barreiras. Tudo se torna fácil e seguro quando se pode aplicar o conhecimento comprovado em desktop. Principalmente para o desenvolvedor que utiliza o IDE do Delphi, os primeiros contatos com as soluções do mercado, desestimulam o profissional logo ao perceber que é preciso uma série de linguagens, frameworks, conceitos e tecnologias o que tornam um investimento alto e uma alta curva de aprendizado. Já o primeiro contato com Morfik é algo extraordinário, pois seu modo de construir soluções web dá ao profissional uma sensação de liberdade, produtividade e um resultado imediato. Nos dias de hoje, precisamos focar na solução para nossos produtos e nada melhor do que utilizando a melhor linguagem, "a que dominamos", por isso, Morfik é a única solução que irá nos atender em todos os requisitos.

apresentação) e na lógica da aplicação em projetos separados. Isso também pode ser conseguido usando aplicações intermediárias em Morfik que consomem serviços publicados por outros aplicativos, independentemente de eles serem construídos com Morfik ou não.

Nota do DevMan

XML (eXtensible Markup Language) é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais. É um subtipo de SGML (acrônimo de Standard Generalized Markup Language, ou Linguagem Padronizada de Marcação Genérica) capaz de descrever diversos tipos de dados. Seu propósito principal é a facilidade de compartilhamento de informações através da Internet. A principal característica do XML, de criar uma infraestrutura única para diversas linguagens, é que linguagens desconhecidas e de pouco uso também podem ser definidas sem maior trabalho e sem necessidade de serem submetidas aos comitês de padronização.

Uma ótima ferramenta para aplicativos de servidor

Com Morfik 3.x você também pode ir ao outro extremo de desenvolvimento web e criar um aplicativo que publica apenas serviços web, sem ter qualquer interface própria. Podem ser usados para implementar um conjunto de serviços que serão consumidos por outros aplicativos. Morfik é projetado para cobrir todas as necessidades básicas de um desenvolvedor que trabalha na criação de uma aplicação web.

Mobilidade Web

Morfik disponibiliza explicitamente a criação de aplicativos para acesso à Web a partir de dispositivos como iPhone, iPod Touch e do iPad. Os desenvolvedores podem criar interfaces diferentes para cada dispositivo no próprio IDE do Morfik no mesmo projeto e decidir quais páginas servirão como base ao serem executadas em um dispositivo mobile.

O suporte para visualização de aplicações em dispositivos móveis é previamente observado durante a depuração por meio de emulador próprio, tornando mais fácil para os desenvolvedores para ver como as suas aplicações ficarão sem ter que publicar constantemente no dispositivo móvel.

Ferramenta para o design

Morfik 3.x é uma ferramenta muito completa para os designers. Morfik em seu próprio ambiente de desenvolvimento permite troca instantânea de temas e estilos pré-definidos ou customizados em tempo de desenvolvimento. O trabalho de um profissional de designer pode ser tranquilamente feito sem a necessidade de outra ferramenta.

Outro desenvolvedor que não tenha experiência com este tipo de tarefa, também é beneficiado com as facilidades encontradas nos recursos de designer. Ambos desenvolvedores obtêm uma imagem clara de como os componentes dos formulários serão mostrados dentro do navegador em tempo de projeto. Isso significa menos preocupações e alterações de projeto para obter a aparência desejada para suas páginas.

Além da Web

A capacidade de importar facilmente Web Services e design sofisticado de interfaces baseadas na web faz Morfik um instrumento privilegiado para a criação de aplicativos que agregam informações de várias fontes. Aplicações Morfik podem facilmente consumir uma infinidade de serviços disponíveis na web e também acessar bancos de dados internos de softwares, reunindo dados de várias fontes em um único aplicativo front-end. Morfik pode estender ou atuar como front-end de aplicativos legados, ou simplesmente para adicionar recursos RIA/Ajax em suas aplicações web-based.

Nota do DevMan

AJAX (acrônimo em língua inglesa de Asynchronous Javascript And XML) é o uso metodológico de tecnologias como Javascript e XML, providas por navegadores, para tornar páginas Web mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações. AJAX não é somente um novo modelo, é também uma iniciativa na construção de aplicações Web mais dinâmicas e criativas.

Alta produtividade

Logo no primeiro contato com Morfik, os desenvolvedores conseguem chegar a níveis sem precedentes na criação de aplicações web sofisticadas. Graças à combinação de compiladores e ambiente de trabalho visual, os desenvolvedores podem rapidamente e facilmente criar aplicações atraentes RIA/Ajax que fornecem sites ou mesmo protótipos ainda totalmente funcionais para projetos. Morfik vem melhorando a produtividade a cada nova versão.

As características de design, como temas e estilos permitem que os desenvolvedores mudem rapidamente a aparência de uma aplicação ou Web Site sem exigir mudanças de códigos. Ainda na parte de código, quanto a utilização de bibliotecas conhecidas em Delphi e também a reutilização de bibliotecas DLLs, facilitam ainda mais todo o desenvolvimento.

Implantação do aplicativo Web

Para cada solução criada no Morfik, pode existir uma ou mais formas específicas de publicação, seja ela um simples site ou uma aplicação complexa com banco de dados ou integrada com outras aplicações de servidor. Vejamos na **Tabela 1** as formas encontradas para atender sua necessidade.

Toda tarefa de servidor e implantação de uma aplicação do Morfik contém um ou mais assistentes para atender desde uma estrutura básica quanto um servidor avançado. Não é necessário um servidor especial para se ter uma aplicação publicada, quando serviços da aplicação exigir acessos especiais a pastas e banco de dados, dependendo da aplicação, pode-se contratar um serviço dedicado ou até uma VPN, no entanto, qualquer servidor de hospedagem do mercado poderá ser compatível com sua aplicação

O ambiente de desenvolvimento

O ambiente de desenvolvimento Morfik foi criado principalmente para acelerar o processo de construção de aplicações web

interativas. Os desenvolvedores podem codificar com as facilidades das linguagens de alto nível, como C #, Pascal e Basic, enquanto o compilador do Morfik irá gerar todo o código JavaScript, HTML e até mesmo muitos dos elementos visuais necessários para implementar o aplicativo. Implementações na forma de widgets (semelhante ao TFrame do Delphi) e pacotes podem ser conectados facilmente adicionar função para sua aplicação, bem como a utilização de temas que fazem com que seja fácil de atualizar e personalizar sua aparência.

Necessidade	Descrição
Implantação do Windows	- xApp stand-alone - Apache 1.3 - Plug-in para Apache 1.3 - Apache Plug-in Native xApp com o Apache 2.2 - Plug-in do IIS xApp com o IIS 6 - Plug-in do IIS xApp com o IIS 7
Implantação Linux	- xApp stand-alone - Apache 1.3 - xApps nas nuvens na Amazon - xApps como um módulo do Apache no Linux nativo
Automação dos Serviços de Implantação	- MorfikSpace Active Directory Services - Software de Servidor de Auto-Implantação - Assistente Morfik de Implantação

Tabela 1. Implantação do aplicativo Web

Na verdade, a única coisa que tem a fazer é entender um pouco sobre as ferramentas do Morfik, isto é muito simples uma vez que tudo é bastante intuitivo. Uma vez que você já passou dessa fase, você conseguirá fazer coisas que nunca conseguiu sem ele.

Construindo a primeira aplicação

Será apresentado o exemplo de construção de uma aplicação de contas a pagar online que permite que os usuários cadastrem suas contas e façam os respectivos lançamentos para que tenham um controle sobre datas e valores de pagamento. Estas tarefas requerem apenas duas tabelas (para armazenar as contas e os lançamentos) e as interfaces de usuário para manutenção das mesmas.

Para criar um projeto, basta escolher a função pelo menu ou botão de ação "New Project". A partir da lista de modelos disponíveis pode-se escolher um modelo que se adapte melhor ao projeto desejado. Se for marcada a opção "Create Sample Pages", o novo projeto irá incluir um número de páginas de amostra (incluindo páginas para dispositivos portáteis). Estas páginas também podem ser adicionadas depois que o projeto é criado usando o comando "New Page".

Neste momento Morfik já tem o suficiente para que seja possível ver algo real funcionando, basta clicar em compilar (F9) e Morfik abrirá o "Morfik Debug Browser" (**Figura 1)**.

Criando tabelas para uma aplicação

Para cada novo projeto, Morfik cria um banco de dados padrão Firebird embedded, com o nome da aplicação e utilizando a extensão "Morfik Database File (.Mxd)", isto pode ser configurado posteriormente utilizando outra conexão e até outro banco de dados como SQL Server, MySQL etc.

Figura 1. Morfik Debug Browser

Nota

No IDE do Morfik, a barra de ferramentas muda dinamicamente, dependendo do contexto. Se você tem um projeto aberto, o botão "New Project" não será visível. Fechar o projeto, ou acessar o comando New Project ou assistente para Novo projeto são acessados por meio do botão Menu Inicial.

Nota

Em sistemas protegidos por um Firewall, o "Windows Security Alert" é exibido no momento da execução do projeto ou pré-visualizar um formulário. Você não deve liberar este aplicativo para que possa rodar seu site.

Com o Assistente do Morfik é possível criar as tabelas para armazenar as informações sobre as contas e outras para lançamentos (pensando no cenário de exemplo). A **Figura 2** mostra onde se encontra o botão de criação de novas tabelas na barra de ferramentas do Morfik.

Figura 2. Morfik Table Wizard

Seguindo o wizard, temos uma interface para criar todos os campos e suas definições desta tabela, veja na **Figura 3**, onde definimos os campos necessários para esta tabela conforme definição da **Tabela 2**.

Assim que todos os campos forem criados, Morfik ainda nos questiona se a tabela irá conter uma chave primária, isto é, a chave única obedecida pelo campo determinado. Uma tela é apresentada

Tabela Contas						
Name	Type	SubType	Description	Default value	Size	Required
Codigo	Number	Integer	Código da conta	0		True
Descricao	Text	Variable Size	Descrição da Conta		40	True
Tipo	Text	Variable Size	Tipo da Conta	P	1	True

Tabela 2. Estrutura da tabela "Contas"

oferecendo a opção de escolher os campos que serão chave primária, ou de ignorar e prosseguir sem essa definição.

Agora basta finalizar o assistente onde ainda será aberta a tabela no modo estrutura para lançamento de valores ou mesmo reestrutura da mesma, caso algum campo não tenha sido criado corretamente, perceba na **Figura 4** este ambiente que permite outras operações como definição de índices entre outras ações que podem ser redefinidas conforme a necessidade do projeto.

No modo DataView (tecla [F12]), basta responder "Sim" se for solicitado uma confirmação para salvar a tabela. É importante notar a existência de controles de navegação para as operações de registros. Para adicionar um registro fictício para a tabela contas (para fins de teste), bastar inserir o primeiro registro com os valores:
Código=1;
Descrição=Supermercado;
Tipo=P.

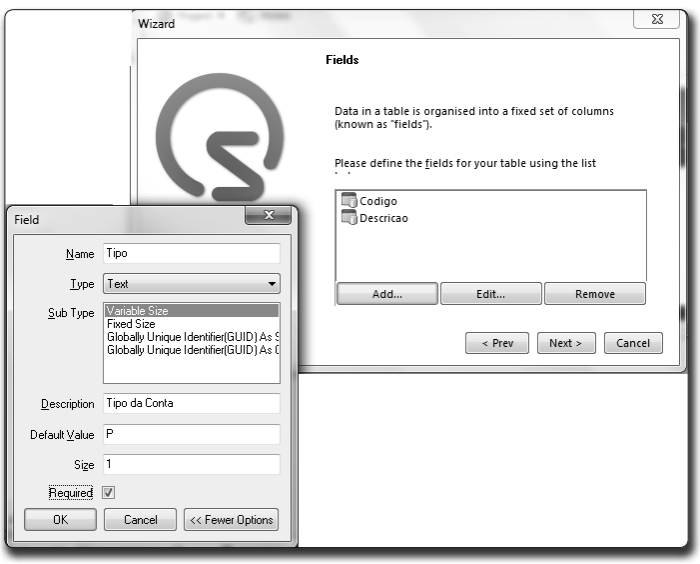


Figura 3. Morfik Table Wizard – Definição dos campos

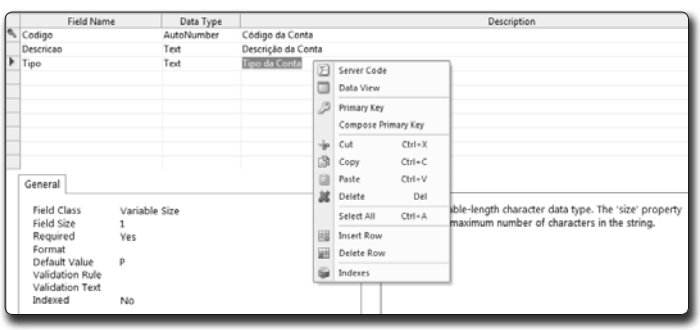


Figura 4. Morfik Table – Modo estrutura

A segunda tabela será a tabela de lançamentos, esta deverá conter a estrutura apresentada na **Tabela 3** e será desenvolvida com o mesmo assistente.

Um novo registro poderia ser inserido, utilizando o modo Data View, com os valores de exemplo a seguir:
Codigo=1;
Conta=1;
DataLancro=20/05/2011;
DataVencto=20/06/2011;
Quitado=N;

Concluindo a criação das tabelas, tem-se o suficiente para uma aplicação funcional.

Nota

Observando cada opção dos assistentes de criação dos objetos na IDE do Morfik, serão encontradas as facilidades para criação de tarefas repetitivas para se ter o máximo de aproveitamento da ferramenta.

Criando os formulários para manipulação dos dados

Qualquer trabalho na interface do usuário do projeto pode começar a partir de uma página que é criada como parte do modelo selecionado. Antes de continuar, vale observar como são formadas as páginas para que se possa entender o estilo de navegação de um site oferecido pelo Morfik.

Abrindo pela Guia da área de trabalho do IDE, encontram-se os dois modelos que foram criados para o site de exemplo. O primeiro é a página Index, esta é uma página preparada para navegação nos navegadores estilo Desktop e a outra Index_iPhone é um template para elaboração do mesmo site sendo navegado em outros dispositivos portáteis. Isto é feito automaticamente pelo Morfik.

Pelas opções do projeto pode-se direcionar qual página deve corresponder para cada plataforma onde o site irá rodar. A **Figura 5** exibe onde são configuradas as opções do projeto.

Ao se definir esta navegação, Morfik irá compilar a aplicação com um código para o navegador que irá identificar o dispositivo e direcionar imediatamente a pagina correspondente. Sendo assim, toda camada visual pode ser feita independente e toda implementação de código ou dados estão protegidas e principalmente livres de implementações web como CSS e javascript.

Antes de construir os formulários, primeiro vamos entender como funciona a camada definida para elaboração das camadas de navegação do site. Quando estamos trabalhando com páginas, a parte de uma página que faz parte do topo (o nível mais alto da interface do usuário) é separada (marcado com 1 na **Figura 6**). As

partes restantes da página são divididas em seções usando sub-formulários (marcado com 2) com o sub-formulário que contém um formulário e a terceira parte (marcado com 3) é destinada ao rodapé da aplicação. Este modelo se assemelha com as interfaces no padrão MDI como são feitas em aplicações em Delphi. Formulários são a tela na qual é possível fazer a maior parte do trabalho de criação de interface de aplicação visual.

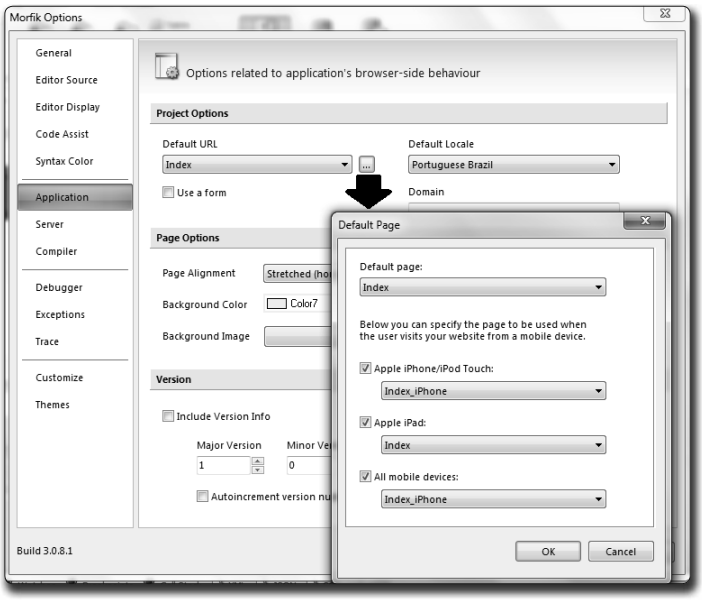


Figura 5. Morfik Options – Definição de navegação por dispositivo

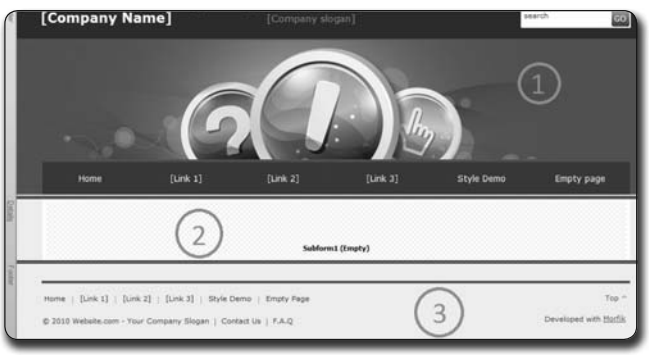


Figura 6. Camadas visuais do site

Tendo isto em mente, aproveitamos então para alterar as legendas dos componentes, bastando para isso clicar com o botão direito no controle e escolher a opção "Edit Caption...". Mais um recurso do Morfik pode ser observado neste instante, cada objeto selecionado tem seu menu de contexto acessado no canto superior direito, este menu irá lhe fornecer as propriedades comuns de cada objeto selecionado. Exemplo: um EditText terá o atalho para Edit mask (definição de mascara de entrada de texto), enquanto um Button terá o atalho para Select Icon (Seleção de ícone para o botão). As demais propriedades de um objeto são encontradas no Object Inspector (F11).

Agora sim iremos elaborar os formulários, partindo com o assistente de criação de formulários, mostrado na **Figura 7**.

Figura 7. Assistente de criação de formulário

O primeiro formulário será o frmContas. Após a definição do nome, definimos para que este contenha uma banda de cabeçalho e outra de rodapé para nosso modelo, escolhendo as opções "Yes, include a header band" e "Yes, include a footer band".

Na sequência, escolhemos o modo datasource e a tabela que será manipulada por este formulário. Veja na **Figura 8** como isso pode ser escolhido.

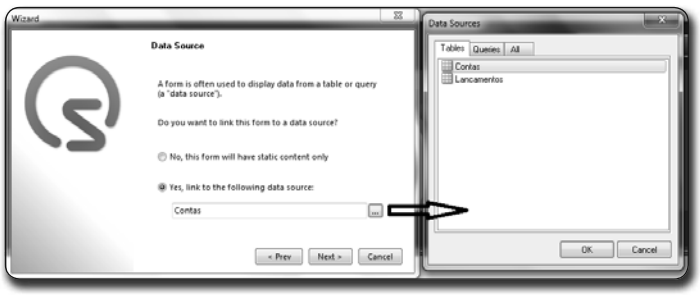


Figura 8. Assistente de criação de formulário seleção de datasource

Tabela Lancamentos						
Name	Type	SubType	Description	Default value	Size	Required
Código	Number	Integer	Código Lançamento	0		True
Conta	Number	Integer	Código da Conta			True
DataLancto	DateTime	Date	Data de lançamento			True
DataVencto	DateTime	Date	Data de Vencimento			False
Quitado	Text	Fixed Size	Quitado S/N	N	1	True

Tabela 3. Estrutura da tabela "Lancamentos"

A próxima etapa é para que seja escolhido o modelo de visualização e consequentemente manipulação dos dados da tabela no formulário. O Morfik oferece três modos de navegação. São eles: Single Form (Layout para um registro por vez); Continuous Form (Navegação estilo Grid, porém no modo somente leitura, apenas agrupamentos e ordenação são suportados) e também Continuous Editable Form (Navegação no modo Grid com edição total, agrupamento e ordenação dos registros).

Para o exemplo do artigo, utilizou-se o modo Single Form (Layout para um registro por vez), no entanto fique a vontade para criar outros formulários com outros tipos de navegação.

A etapa seguinte do assistente de formulário nos dá opção para que seja adicionado um controle de navegação podendo ainda ser totalmente customizado, isto é feito conforme a **Figura 9**.

Figura 9. Assistente de criação de formulário barra de navegação

Para os controles baseado na estrutura da tabela, o assistente de formulário, analisa cada campo e permite a escolha de quais os campos deverão aparecer no formulário e ainda quais os tipos de controles irão representá-los. Na **Figura 10** temos uma visão clara deste recurso e já podemos escolher o controle Combobox para o campo tipo que terá os valores P(Pagar) e R(Receber) que veremos mais tarde.

Concluído o assistente de criação de formulário, temos o frmContas onde já é possível efetuar algumas alterações, conforme a **Figura 11**. Temos tudo praticamente construído e assim podemos atribuir os valores dos itens para o campo do tipo combobox escolhido durante as etapas do assistente.

Nota

Embora seja fácil e rápido desenvolver toda a aplicação utilizando os assistentes da ferramenta, é importante que o desenvolvedor procure criar os mesmos objetos da aplicação no modo manual, isto irá facilitar o entendimento e maior controle sobre as camadas.

Figura 10. Assistente de criação de formulário definição de controles

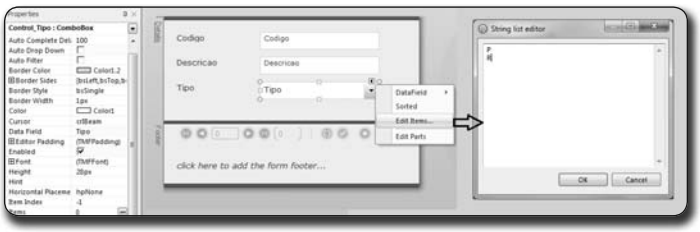


Figura 11. Alterando controles do formulário

Seguindo o mesmo assistente de criação de formulário, outro formulário deve ser criado para manipulação da tabela lançamentos, para exemplo, o nome de frmLancamentos será usado, e ao chegar na etapa de controles deve-se alterar os controles para os campos Conta e Quitado para o tipo de controle ComboBox (idêntico a **Figura 12)**. O assistente deve ser seguido até o final para que se possa realizar alguns ajustes para o cadastro.

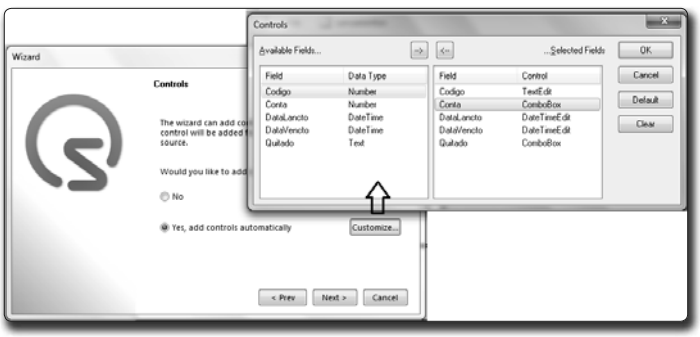


Figura 12. Assistente de criação de formulário para tabela Lançamentos

Agora com os formulários criados, é permitido realizar o ajuste do controle Conta do frmLancamentos para que se comporte igual a um controle de lookup, isto é, a referência deste valor está em uma tabela externa, onde será armazenado o código de uma conta e exibido a descrição da mesma no controle. A técnica é bem simples e pode ser observada na **Figura 13**. Configure exatamente como mostrado e terá o funcionamento perfeito para o cadastro de um lançamento de uma conta no site.

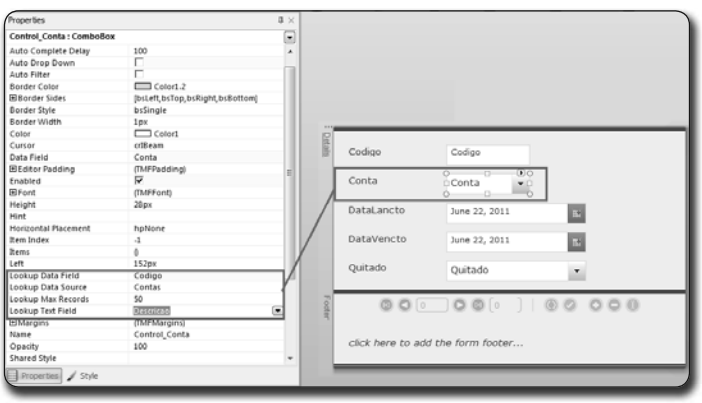


Figura 13. Configuração de campo lookup para formulário de lançamentos

No mesmo formulário, os itens S e N devem ser adicionados ao campo Quitado representado pelo controle ComboBox.

Nota

Em caso do uso de lookup, devem-se tomar cuidados quanto a desempenho principalmente em aplicações web. Prefira construir consultas otimizadas e realizar testes de desempenho para este tipo de implementação.

Criando um menu

Os componentes dos formulários têm suas propriedades e eventos conforme o tipo, ou seja, um Button tem eventos como onClick semelhante ao Delphi, Morfik nos fornece assistente de códigos que irão basear em ações padronizadas e podem ser a base para simples chamadas de formulários e também aplicação de regras de negócio disparadas por métodos dos eventos dos componentes. A **Figura 14** ilustra a sequência de ações necessárias adicionar um novo evento para a chamada do formulário de contas pelo evento onClick do botão contas.

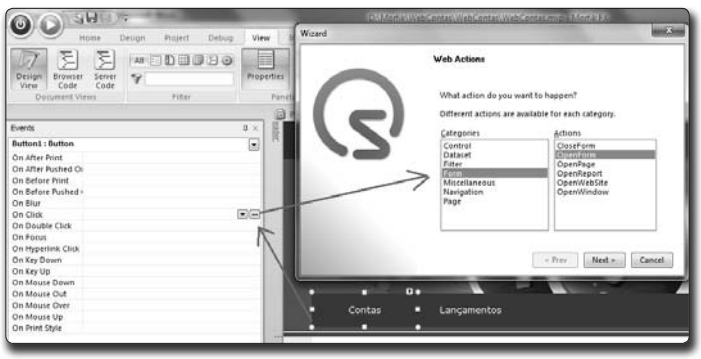


Figura 14. Ações para eventos dos componentes

Este assistente leva a definição de qual o formulário deseja-se abrir com esta ação e onde este será exibido, no caso frmRoot:Subform1 que é o Subform principal do site. Os parâmetros vistos junto a **Figura 15** são utilizados para se ter uma comunicação entre formulários e/ou passar um filtro para o datasource.

Finalizando o assistente de código e indo para a visualização de Browse Code (Código de navegador). Veremos a implementação conforme **Listagem 1.**

Listagem 1. Exemplo de código gerado pelo assistente

```
1. Procedure frmRoot.Button1Click(Event: TDOMEvent);
2. Begin
3.   OpenForm('frmContas', 'frmRoot:Subform1', '');
4. End;
```

Retorne para a página principal do site e utilize o assistente de código para criar a chamada para o frmLancamentos. Finalmente execute sua aplicação e faça seus testes cadastrando suas contas e seus lançamentos. Veja na **Figura 16** como ficará o site.

Procure utilizar os demais assistentes para ir se familiarizando com Morfik e construir suas próprias aplicações. A dica final para este exemplo é para implementação de regras de cadastro utilizando o evento conforme a **Listagem 2**.

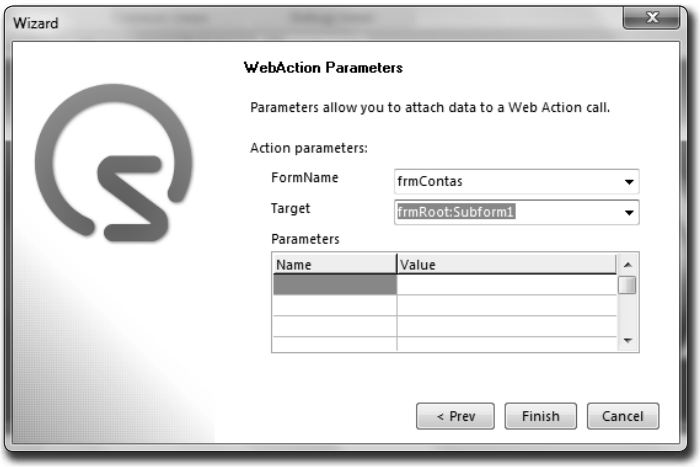


Figura 15. Escolhendo o formulário a ser aberto pela ação

Figura 16. Testando a aplicação

Listagem 2. Validando campos requeridos no envio do formulário

```
1. Procedure frmLancamentos.WebFormBeforeSubmit(Var Cancel: Boolean);
2. Begin

3.   if ( Control_Codigo.Text = '' ) or (Control_Conta.Text = '') then
4.   begin
5.     Cancel := True;
6.     ShowMessage('Atenção, Campos requeridos não foram preenchidos');
7.   end;

8. End;
```

Resumo dos principais recursos e ferramentas do Morfik

Acesso a banco de dados - Morfik tem suporte nativo ao banco de dados Firebird 2.x, pode se utilizar um banco de dados embedded ou mesmo criar/reutilizar de outra conexão. Também podem ser feito acesso por meio de conexões ODBC, OLEDB, Access, Excel, XML ou TXT. Além destes acessos, existem pacotes como o Zeos Database Free que pode ser instalado no Morfik onde irá se trabalhar com conexões com banco de dados MSSQL, PostgreSQL ou MySQL nativamente.

Consumo e publicação de serviços Web - São suportados os principais meios de comunicação de serviços web, por meio de consumo de Web Services, SOAP ou REST/JSON.

Nota do DevMan

REST é a abreviação de Representational State Transfer ou Transferência de estado representacional. Ele surgiu da tese de doutorado sobre a Web escrita por Roy Fielding, que foi um dos idealizadores do protocolo HTTP.

Internacionalização - Projetos criados com Morfik tem suporte fácil a localização e internacionalização sem necessidade de reengenharia, qualquer objeto do site pode ser adaptado de uma única maneira, isto é, sem código ou biblioteca adicional.

Criação e publicação de relatórios PDF – A criação de relatórios é tarefa com o mesmo conceito de se criar um formulário ou página do projeto, com isto, novamente Morfik se destaca com a produtividade e recursos que estarão disponíveis instantaneamente para o cliente, a técnica utilizada é a mesma encontrada em muitos geradores do mercado, com a vantagem de estar presente em um único projeto.

Monitoramento e rastreamento de aplicação - Podemos definir monitores para rastreamento de solicitações e respostas http, mensagens SOAP/XML/JSON, enviadas e recebidas pelo servidor, monitoramento de instruções SQL direto no servidor por meio de simples configurações no projeto.

Nota do DevMan

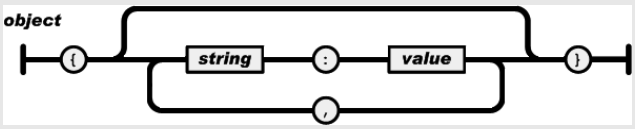
SOAP (originado do acrônimo inglês Simple Object Access Protocol) é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída, utilizando tecnologias baseadas em XML. Sua especificação define um framework que provê maneiras para se construir mensagens que podem trafegar através de diversos protocolos e que foi especificado de forma a ser independente de qualquer modelo de programação ou outra implementação específica.

Conclusão

Com o conceito de Morfik, podemos tranquilamente repensar nosso modo de construir aplicativos web e acelerar nossos investimentos com soluções baseadas em negócios e não em linguagens ou infraestrutura. Diferente do foco das ferramentas do mercado atual, Morfik pode antecipar o futuro das soluções web, uma vez que já podemos levar grande parte da experiência desktop que temos em mãos.

Nota do DevMan

O JSON (JavaScript Object Notation) é uma espécie de padronização para transporte de informações entre ambientes heterogêneos e que tem como linguagem o Javascript. Bem, padrão, transporte, ambiente heterogêneo, tudo isso nos remete ao XML, então vem a questão: Por que JSON e não XML? O principal motivo seria pelo fato da notação JSON ser extremamente mais simples do que a XML. Não existem tags diversas nem regras complexas para escrita e muito menos mecanismos complexos de parser. Para se ter uma ideia, a figura a seguir, ela mostra a sintaxe para notação JSON para um objeto.



Carlos Eduardo Paulino
cadumicro@hotmail.com
http://www.morfikbr.wordpress.com
Desenvolvedor de sistemas há 9 anos, trabalhando na Htec Consultoria e Informática com soluções para o terceiro setor, software jurídico e customizações em ERP Domínio em Delphi, Morfik, Flex e .NET. Praticante de metodologias ágeis, utilizando Scrum e XP.

MorfikBR
http://www.morfikbr.wordpress.com

Instalando o Morfik
http://www.morfik.com/#downloads/Software/Morfik 3

Pacotes e widgets para o Morfik
http://www.morfik.com/#downloads/categories/Packages

Site oficial do Morfik
http://www.morfik.com

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Bancos de Dados Gratuitos – Parte 2

Vitor Luiz Rubio

No artigo anterior foram apresentados os bancos de dados open-source MySql, Firebird e PostgreSQL. Foram apresentados componentes alternativos para acesso aos dados e bancos de dados de exemplo foram testados. Também foram feitas medições de tempo em "selects" com várias tecnologias diferentes.

Porém apenas a performance da consulta foi medida, dos bancos de dados em si não foi medida a performance por não termos um banco de dados de teste igual para cada SGBD.

Microsoft SQL Server Express

O Microsoft SQL Server Express sem dúvida é um dos mais conhecidos e queridos sistemas gerenciadores de bancos de dados, principalmente por programadores que não trabalham apenas com Delphi. Tem uma confiabilidade típica de bancos de dados profissionais, além de um controle rígido de acessos, usuários e papéis. Esses devem ser muito bem configurados para que seja implementada uma boa segurança. Outras características interessantes do SQL Server são suas stored procedures, cuja sintaxe lembra um pouco o pascal.

Um ponto muito interessante no SQL Server é que, embora ele tenha recursos avançados de gerenciamento, automação e conversão, que outros bancos de dados gratuitos não têm, ele não se torna dificílimo de aprender por causa disso. Um recurso muito interessante é o FullTextSearch, onde é possível eleger tabelas para criar catálogos de palavras/strings e poder fazer uma busca com a função "contains", que aceita muitos parâmetros e coringas diferentes, e é muito mais eficiente do que um "like" tradicional.

Na seção de links desse artigo encontra-se a URL para download do SQL Server Express. O processo de instalação é um pouco demorado, mas não tem segredos. A **Figura 1** mostra o início de sua instalação.

Nota do DevMan

FullText Search

O tipo de consulta categorizado como FullText é o tipo que cria um índice com as palavras de um texto. Uma vez que esse índice é criado, pode-se efetuar consultas no estilo utilizado por buscadores (Google, Bing). No geral, o texto a ser indexado passa por um analisador, que tem a função de identificar palavras chave, passando essas palavras para o indexador, que mantém isso em ordem. Esse recurso não é exclusivo para bancos de dados relacionais como SQL Server, Oracle ou MySQL. Existem também os frameworks de FullText Search. O mais conhecido na área de desenvolvimento de software é o Lucene (para Java) e sua versão para .Net. A principal vantagem em se usar um framework externo é retirar essa responsabilidade do banco de dados, deixando-o livre para realizar as consultas principais de um sistema.

Resumo DevMan

De que trata o artigo:

Este artigo é uma continuação do artigo anterior e trata sobre bancos de dados gratuitos, tanto proprietários como open-source, e das maneiras disponíveis no mercado de se conectar a eles. Serão apresentados os bancos de dados, o passo a passo para sua instalação e os componentes de conexão.

Em que situação o tema é útil:

A escolha de um sistema gerenciador de banco de dados é um passo crucial no planejamento e análise de um sistema. Além disso, muitos sistemas têm a necessidade de acessar mais de um banco de dados diferente, e o desenvolvedor deve saber acessar todos. O SQL Express e o Oracle Express são as versões gratuitas, e limitadas, dos mais robustos e populares bancos de dados proprietários.

Bancos de dados Gratuitos:

Não é apenas de Interbase e Firebird que o Delphi vive. No artigo anterior foram abordados, além do banco de dados Firebird, os bancos MySql e PostgreSQL. Neste artigo serão mostrados o banco de dados Microsoft SQL Server 2008 Express e o Oracle 10G Express e o IBM DB2. Apenas esses seis bancos de dados não cobrem nem de longe a totalidade dos bancos de dados existentes. Além disso, BDs relacionais não são o único tipo de banco existente. Em artigos futuros serão abordados outros tipos de bancos de dados.

O instalador do SQL antes verificará se todos os pré-requisitos estão instalados. Caso falte algum requisito o software acusará. Depois de instalados todos os requisitos clique em re-run. Então pode-se escolher quais recursos instalar. O SQL Server trabalha com o conceito de instância de servidor, assim, durante sua instalação deve-se informar um nome ou utilizar o sugerido, como mostra a **Figura 2**.

Um usuário deve ser escolhido para ser o usuário no qual o SGBD irá rodar. Geralmente é criado um usuário com o mesmo nome do banco de dados (o PostgreSQL e o MySQL fazem isso automaticamente, já o SQL Server permite que você crie ou escolha). Pode-se escolher um dos usuários de sistema do Windows para a execução do sistema gerenciador de banco de dados. Na sequência da instalação escolhe-se o tipo de autenticação (Authentication Mode).

O modo de autenticação "Windows" permite que o usuário, no caso o programa que acessará o banco ou o próprio SQL Management Studio, entre com as credenciais do Windows, ou seja, o login

e senha atuais ou um login e senha da rede, do Active Directory ou da máquina servidora. Já o modo "mixed" permite o uso de usuários e senhas do próprio SGBD além das credenciais do Windows.

Figura 1. Clicar em "New Instalation" para iniciar o processo

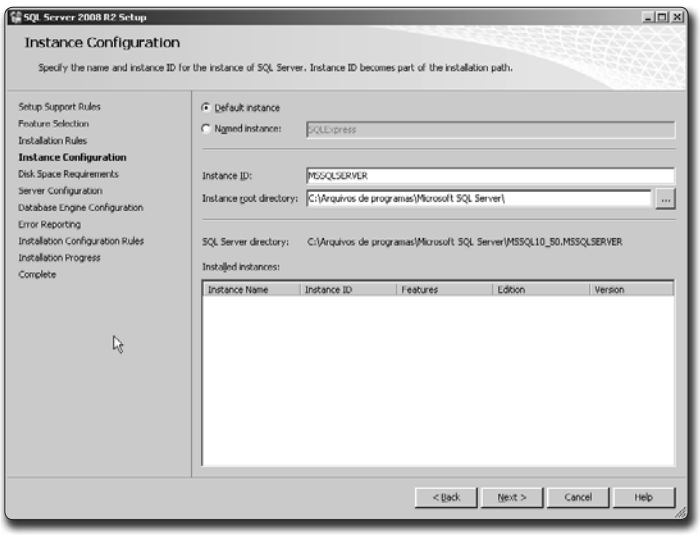


Figura 2. Escolha do nome de instância

Nota do DevMan

Grandes empresas possuem dezenas, quando não, centenas de máquinas conectadas em rede. A localização de um determinado recurso (impressora, scanner, estação) compartilhado precisa ser rápido e fácil. Para gerenciar isso, em ambientes Windows, a Microsoft desenvolveu o Active Directory. Este é um serviço de rede que cria uma banco de dados centralizado com as informações da rede, facilitando a localização de qualquer recurso. Foi implementado utilizando protocolos padronizados como o LDAP (Lightweight Directory Access Protocol) o que permite um intercâmbio de informações com outros produtos que também utilizam o padrão LDAP

O Microsoft SQL Server possui vários bancos de dados de exemplo. Alguns mantidos pela própria Microsoft e outros mantidos pela comunidade. O interessante dos bancos de dados de exemplo da Microsoft é que por trás deles há uma "historinha" como se fossem os problemas e levantamentos de requisitos de uma empresa fictícia e todo o material didático dos treinamentos Microsoft é baseado nessa empresa "de mentirinha". Destaque para o banco de dados AdventureWorks que se trata de uma fábrica de bicicletas. Na seção de links há a url para baixar os bancos de testes da Microsoft. A conexão pode ser realizada com um desses bancos.

Uma vantagem do banco de exemplo Northwind da comunidade é que ele pode ser recriado do zero a partir da execução de um arquivo .sql que pode ser baixado na seção de links e também está disponível com o fonte desta edição.

Os valores dos inserts podem ser mudados para que dados diferentes sejam inseridos no banco. Fique à vontade para modificar de acordo com seu gosto.

Para criar o banco de dados basta executar o script sql em uma janela do SQL Server Management Studio (**Figura 3**).

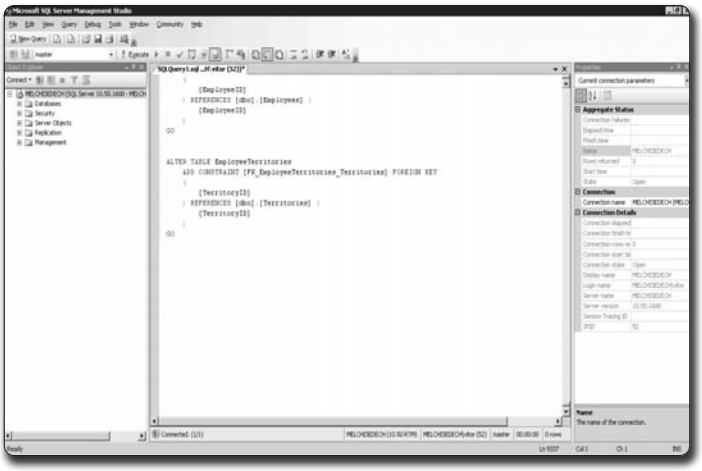


Figura 3. O SQL Server Management Studio

Oracle Express

O mais respeitado banco de dados tem também uma versão Express, e embora configurações finas sejam sempre bem vindas não é necessário ser um DBA para instalar o Oracle 10G.

A única parte da instalação do Oracle que merece uma atenção é a parte onde se coloca a senha para o usuário system, que é o "admin" do Oracle, mostrado na **Figura 4**.

Junto com o banco de dados é instalado um web Server e um aplicativo para gerenciamento do banco. Toda a administração do Oracle, como criação de tabelas, gerenciamento de recursos, memória, usuários e comandos SQL será feita através de um browser.

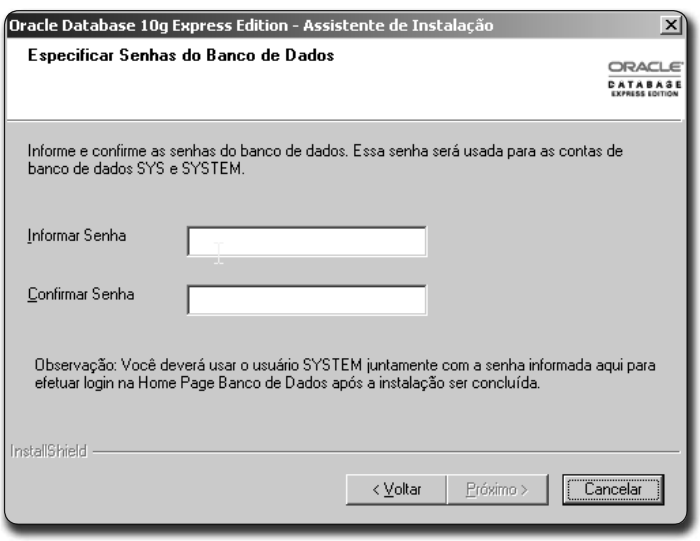


Figura 4. Definição de senha para o usuário SYSTEM do Oracle

O Oracle tem o conceito de instância e schema. O banco de dados (database) é a instância, enquanto que para cada usuário criado ele cria um schema diferente.

O conceito de schema é o que chamamos de database em outros bancos de dados como Firebird. Enquanto que o conceito de instance é o que é chamado de namespaces em outros bancos.

Uma limitação chata do Oracle Express Edition é que ele permite a criação de apenas uma instância. Isso pode não ser considerado uma grande limitação, mas imagine-se criando dois sistemas de domínios completamente diferentes em um mesmo servidor. Cada "usuário" nos arquivos de configuração dos sistemas acessa apenas um "banco de dados", não tem como dois usuários acessarem um mesmo banco, ou como um usuário acessar vários bancos. Na versão full do Oracle não existe essa limitação: é possível criar várias instâncias de banco de dados.

Isso tudo significa que, diferentemente dos bancos de dados vistos até agora, quando se der o preenchimento do parâmetro database ou database name de um componente de acesso a dados deve-se escolher o nome da instância padrão. O login e senha é que determinará na verdade qual é o "banco de dados" ou o conjunto de tabelas que será visível.

Como cada usuário tem um schema, ou table-space, diferente, em dois usuários pode existir tabelas com o mesmo nome.

Além da limitação de uma única instância ele possui outras limitações:

1 – A limitação de executar em apenas um processador, mesmo que o servidor tenha vários;

2 – As tabelas de usuário (user data), que são as tabelas usadas por seu sistema excetuando-se as tabelas de sistema do próprio Oracle, podem chegar ao tamanho máximo de 4gb;

3 - A edição Express pode usar no máximo 1gb de memória livre, mesmo que o servidor tenha mais que isso.

Nota

Depois de instalado o Oracle deve-se decidir se o projeto de teste rodará no mesmo computador que o servidor Oracle ou se rodará em outro. Independente da escolha, no computador onde funcionará a aplicação, que chamamos de client, deve-se instalar as bibliotecas client do Oracle, disponíveis para download na seção de links (arquivo OracleXEClient.exe). Depois disso é necessário configurar os arquivos tnsnames.ora e listener.ora que estão na pasta C:\oraclexe\app\oracle\product\10.2.0\server\NETWORK\ADMIN. A configuração padrão deve funcionar, entretanto verifique a documentação do Oracle para eventuais dúvidas.

DB2 Express

Ao iniciar a instalação do DB2 Express, nos deparamos com a escolha de qual modo de instalação será utilizado. A instalação feita será uma instalação típica. O único recurso que não está presente na instalação típica é o Full Text Search. Para isso faça uma instalação customizada.

Outra configuração delicada é indicar se o serviço rodará com o usuário db2admin ou com a conta LocalSystem do Windows (**Figura 5**). Cada caso é um caso, deve-se analisar isso tanto na concepção do sistema como na implantação, deve-se questionar onde o banco irá rodar, se é em um datacenter, quem fará a administração, quais as vantagens e desvantagens de se usar uma conta do sistema, quais as vantagens e desvantagens de se usar uma conta comum, o que é mais seguro, o que é mais flexível, qual opção permite acesso do banco de dados à arquivos no disco, para importação e assim por diante.

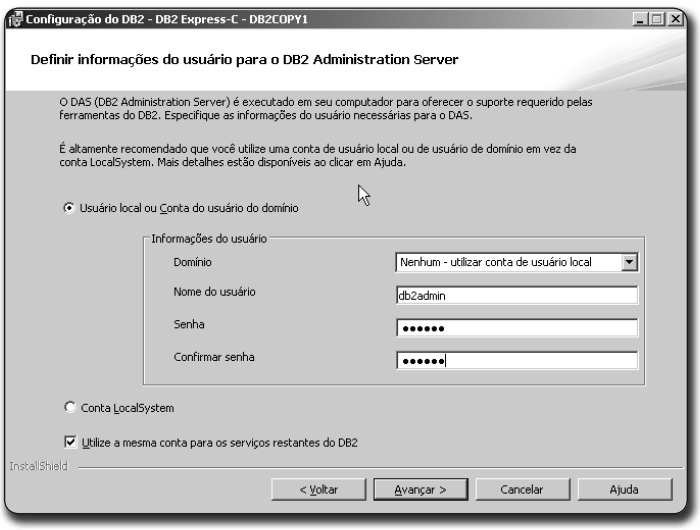


Figura 5. Definição de usuário

A visão do banco de dados que este artigo pretende dar é uma visão do ponto de vista do desenvolvedor, que deve-se preocupar mais com o código e as regras de negócio. Detalhes do banco de dados devem ficar a cargo do DBA especialista na ferramenta selecionada.

No exemplo a senha do usuário db2admin foi "1234567890a_" isso porque as políticas de segurança do Windows ou da rede podem exigir tamanho mínimo de senha e mistura entre caracteres numéricos, alfanuméricos e especiais. Isso pode acontecer durante a instalação do PostgreSQL também. Após a instalação será oferecida a criação de um banco de dados de exemplo, como mostra a **Figura 6**.

Porém, dependendo do tipo de instalação escolhido é possível que o banco de dados SAMPLE já esteja criado. É possível também que tenha sido criado automaticamente um schema com o seu nome de usuário do Windows e que o banco de dados SAMPLE esteja dentro desse schema.

Para ilustrar, no momento em que eu terminei a instalação em meu computador para a criação desse artigo foi criado o schema VITOR e o banco de dados SAMPLE está dentro dele.

Pode ser que o banco de dados SAMPLE já exista e não precise ser criado. De qualquer forma, ao criar o SAMPLE é oferecida a opção de ser criado com objetos em SQL e XML ou apenas SQL. Isso é possível devido ao suporte a armazenamento de XML nativo que o IBM DB2 possui. O suporte a XML não será usado no exemplo desse artigo, então pode-se instalar apenas o SQL.

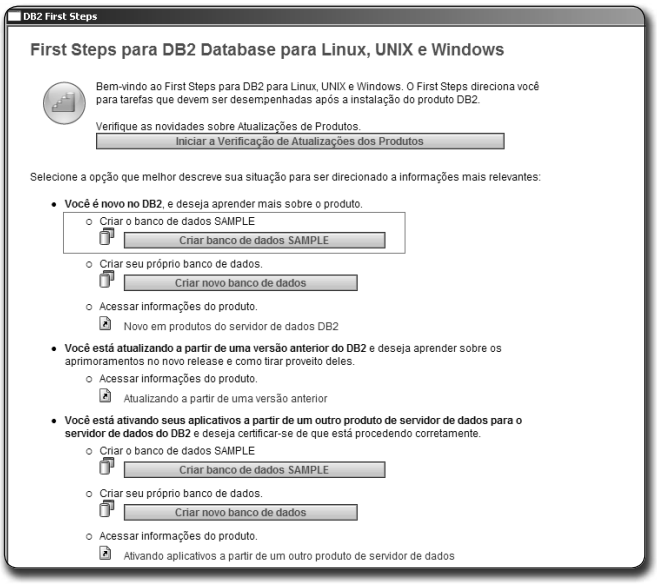


Figura 6. Criação do banco de dados SAMPLE

Ferramentas de administração de bancos de dados

No artigo anterior foi apresentado o PGAdmin, que vem junto com o postgreSQL e é o sistema de administração e consultas ao banco de dados PostgreSQL.

O MySql, embora não venha junto com uma ferramenta visual, possui uma ferramenta modo texto (console). A ferramenta mais usada para a administração do MySql é o PHPMyAdmin, que é uma ferramenta Web e precisa do Apache mais o PHP para funcionar. Toda a administração do banco ocorre através de um browser.

O Interbase possui o IBConsole. Para acessar o Firebird pode-se usar o IBO Console, muito parecido com o IBConsole, ou pode-se usar o IBExpert. Há uma versão gratuita do IBExpert que mesmo tendo menos recursos que a versão paga ainda assim tem bem mais recursos que o IBO Console.

A ferramenta de administração do SQL Server da Microsoft é o SQL Server Management Studio visto na **Figura 3** e a ferramenta de administração do DB2 é o Centro de controle visto na **Figura 7** que pode ser encontrado em IBM DB2>Nome da instância>Ferramentas gerais de administração>Centro de controle.

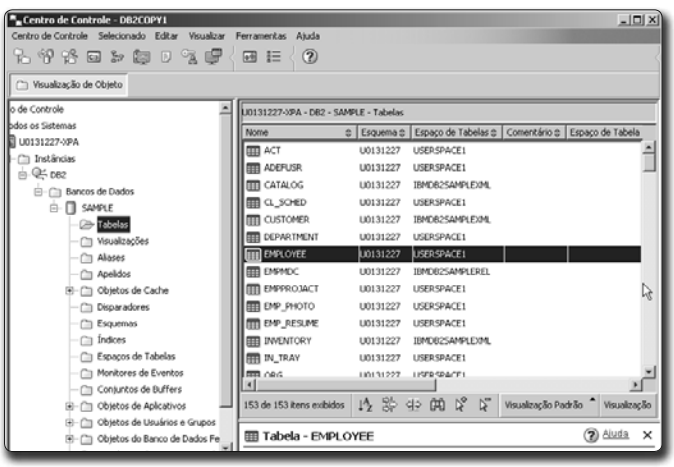


Figura 7. Centro de controle do DB2

A **Figura 8** mostra a ferramenta de administração do Oracle, que funciona inteiramente via Browser com a ajuda de um servidor web embutido no produto. Seu endereço é http://127.0.0.1:8080/apex/ e com essa ferramenta é possível criar usuários/schemas, tabelas, executar comandos SQL e configurar o banco de dados.

Além das ferramentas padrão para cada banco de dados é importante mencionar que existem empresas especializadas em desenvolver aplicações para administração de bancos de dados. Algumas delas são específicas para um determinado SGBD, outras trabalham com múltiplos produtos.

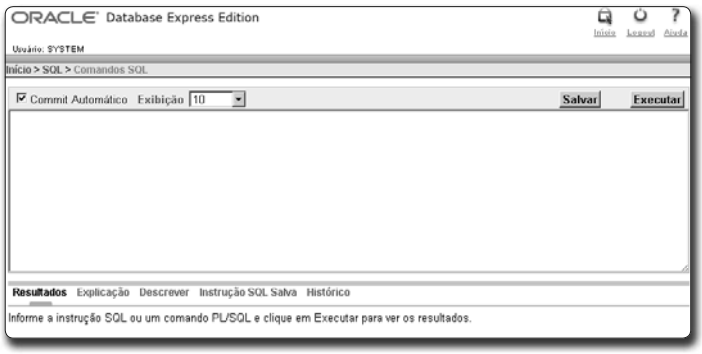


Figura 8. Execução de comandos SQL na ferramenta de administração do Oracle

Conectando-se ao Microsoft SQL Server Express

Para se conectar com o SQL Server tem-se entre as principais opções o DBExpress, preferido por quem visa a independência de banco de dados futuramente, ou pelo menos a substituição do mesmo. Outros desenvolvedores preferem o ADO devido ao fato de ser tecnologia também da Microsoft, portanto, "nativa".

A grande verdade sobre os bancos de dados e as tenologias de acesso é que se forem usadas apenas instruções SQL ANSI padrões e não forem usados recursos exclusivos dos bancos de dados, o tipo do SGBD escolhido não faz diferença. O que faz diferença é o design do modelo de dados, a maneira como esses dados são administrados, armazenados e obtidos.

Há também diferenças de sistema operacional, rede e outros fatores que podem influenciar. Há bancos de dados muito rápidos com poucos dados, mas não muito confiáveis com um volume de dados muito grande. Por outro lado há bancos de dados muito robustos para se trabalhar com terabytes de informação, mas pesados demais para se fazer uma simples agenda.

Devido a esses fatores tenha em mente que nosso teste não é definitivo ou conclusivo. É apenas um teste de talvez centenas que você tenha que fazer. O ideal é que, dependendo do tamanho do projeto, depois de escolhido o banco de dados um especialista na ferramenta escolhida seja consultado.

Outras maneiras de se acessar o SQL Server também incluem BDE com ODBC e Unidac, que continuará sendo testado neste artigo.

Para se conectar via DBExpress deve-se colocar o SQL Connection, SQL Dataset, Dataset Provider e um Client Dataset e se fazer as devidas conexões entre os componentes. Se você é iniciante em Delphi ou nunca trabalhou com DBExpress, a propriedade Connection do SQLDataset deve ser o SQLCon-

nection, a propriedade Dataset do DatasetProvider deve ser o SQLDataset e o ProviderName do ClientDataset deve ser o DatasetProvider colocado.

Nas configurações o tipo de conexão é MSSQLConnection, o driver MSSQL, o login é "sa" ou qualquer outro que tenha sido criado, a senha é a senha escolhida na instalação ou na definição do usuário, nesse caso 123456 e o nome do banco de dados é o northwind.

A instrução SQL no SQLDataset é select * from "Employees". Esta tabela employees tem poucos registros, logo o teste pode não ser muito confiável. Mais uma vez é possível testar com outra tabela, como customer, ou fazer um teste populando a tabela employees com um milhão de registros.

Também faz diferença se esses dados estão em cache ou não. Se eles já foram acessados uma vez com certeza estarão em cache.

A conexão via ADO é praticamente igual, e com dois cliques no componente ADOConnection é possível setar todos os parâmetros da conexão (**Figura 9**).

É possível escolher o servidor a partir de um "combo box". No caso do exemplo Melchisedech é o nome (ou host) do computador onde está instalado o banco de dados. Pode ser localhost se o exemplo rodar na mesma máquina, como foi mostrado no exemplo com DBExpress.

Para o acesso via ADO pode ser usado o driver nativo SQL Server Native Client 10.0 ou o provider OLEDB. Embora a conexão esteja sendo feita diretamente com o ADODataset, este também pode ser ligado a um DatasetProvider e a um ClientDataset. A maioria dos datasets pode.

Com o Unidac não é diferente. Será necessário um UniConnection, um UniTransaction, um UniQuery e um UniProvider do tipo TSQLServerUniProvider. Dois cliques no componente connection abrem um component editor que permite a configuração inteira.

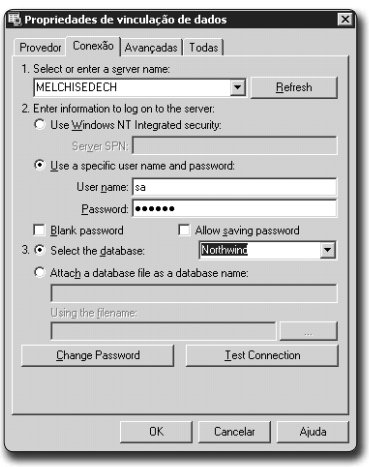


Figura 9. Configuração do ADO para acessar o SQL Server

A **Listagem 1** mostra o código para se abrir o dataset e exibir o tempo que leva para carregar os dados. Lembrando que aqui está sendo usado um único datasource e datagrid para todos os datasets, e é por isso que o dataset do datasource está sendo definido via código, e ligado apenas depois que o dataset esteja aberto.

Listagem 1. Teste com SQL Server

```
procedure TForm1.Button1Click(Sender: TObject);
var
 T: integer;
begin
 if ClientDataSet1.Active then
     ClientDataSet1.Close;
     t := GetTickCount;
     ClientDataSet1.Open;
     t := GetTickCount -t;
     Label1.Caption :=  IntToStr(t);
     DataSource1.DataSet := ClientDataSet1;
end;
procedure TForm1.Button5Click(Sender: TObject);
var
 T: integer;
begin
 if ADODataSet1.Active then
     ADODataSet1.Close;
     t := GetTickCount;
     ADODataSet1.Open;
     t := GetTickCount -t;
     Label2.Caption :=  IntToStr(t);
     DataSource1.DataSet := ADODataSet1;
end;
procedure TForm1.Button6Click(Sender: TObject);
var
 T: integer;
begin
 if UniQuery1.Active then
     UniQuery1.Close;
     t := GetTickCount;
     UniQuery1.Open;
     t := GetTickCount -t;
     Label3.Caption :=  IntToStr(t);
     DataSource1.DataSet := UniQuery1;
end;
```

Conectando-se ao IBM DB2

Conectar-se ao IBM DB2 é tão fácil quanto se conectar ao SQL Server. Pode-se usar inclusive os mesmos componentes, já que também é possível conectar-se ao DB2 com os componentes ADO.

Existe uma diferença sutil: não existe servidor e database, mas no db2 ambos são uma coisa só. Como no exemplo, no momento da instalação do DB2 foi criado um banco de dados de exemplo SAMPLE. Para se conectar a ele, sample é o nome tanto do banco de dados como do servidor. É necessário apenas colocar o nome do banco de dados, sendo que o servidor não é necessário.

Uma exceção é o ADO, que também permite conexão com nome do servidor, porta e nome do banco.

Outra peculiaridade do DB2 é que embora o nome do banco seja o nome do server, ele possui o conceito de tablespaces e schemas, e cada usuário cria automaticamente um schema com o mesmo nome que o usuário. O usuário db2admin, cuja senha foi definida na instalação, talvez não tenha acesso ao banco de dados sample, talvez nem o veja. Se isso acontecer é porque o banco foi criado com o seu user como owner. Sendo assim use o seu login e senha do Windows em vez do login db2admin.

Da mesma forma que nomedobanco.nomedatabela é uma construção válida no SQL Server, no DB2 é possível a construção nomedousuario.nomedatabela, como VITOR.EMPLOYEE. Outra construção válida é nomedobanco.nomedousuario.nomedatabela, como SAMPLE.VITOR.EMPLOYEE. Cada banco de dados pode

ter um ou mais usuários. Cada usuário é/tem um schema ou pode ter mais de um, e cada schema contém tabelas. Sendo assim as tabelas não estão diretamente dentro do banco de dados como é comumente visto nos outros bancos de dados, mas as tabelas estão contidas dentro dos "schemas". Mesmo atribuindo permissão para o db2admin acessar a tabela employee, ele deve acessar VITOR.EMPLOYEE, pois EMPLOYEE está no schema VITOR.

Isso quer dizer que quando o usuário dono do schema está logado ele pode acessar diretamente a tabela employee com um select * from employee. Quando outro usuário está logado, como o db2admin na propriedade username, então deve ser feita a consulta select * from nomedoschema.employee, tendo ele permissão para tal.

No utilitário de configuração/administração do db2 é possível incluir mais schemas para cada usuário, além do que é criado automaticamente, atribuir permissões para acesso a tabelas, criar aliases para tabelas e definir um schema para o alias (apelido), assim a tabela pode ser acessada diretamente sem o nome do schema, através de seu alias e uma série de outras configurações avançadas. O IBM DB2 é um banco bastante robusto e avançado, para aplicações de grande porte.

Nota

Também existe o conceito de schema no MS SQLServer. Dois dos schemas principais são DBO e SYS. Porém o uso deles não se faz tão evidente, e caso o usuário tenha permissões ele pode acessar as tabelas diretamente pelo nome ou com NomeDoBanco.NomeTabela. Além disso, o ADO e o DBX se encarregam de colocar os schemas automaticamente na maioria das situações onde eles são necessários.

Ao se conectar no DB2 pelo DBX pode-se notar que é possível escolher o schema, tanto na janela de properties do componente SQLDataset quanto na sua janela de edição, conforme visto na **Figura 10**. Mesmo assim ainda é necessário colocar o nome do schema antes do nome da tabela, a não ser que se esteja fazendo login com o usuário owner do schema onde a tabela está.

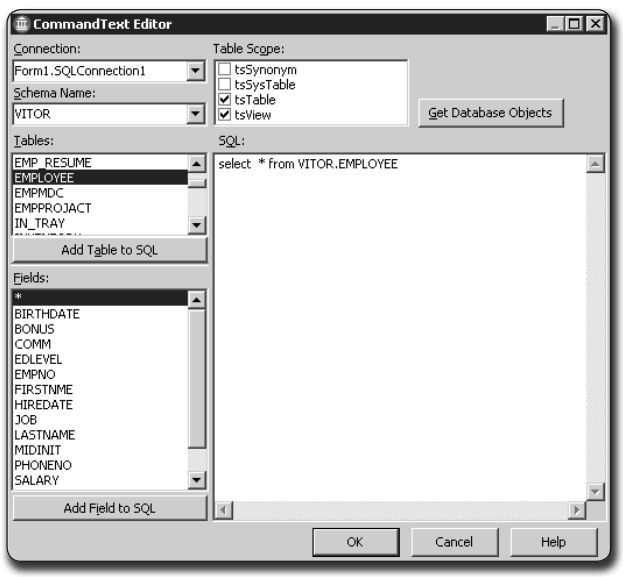


Figura 10. Criação de query no DBX com NomeSchema.NomeTabela

A conexão ao db2 via ADO pode ser de duas formas: especificando-se nome do servidor, porta e nome do banco de dados ou simplesmente preenchendo o parâmetro origem dos dados, que o ADOConnection, com ajuda dos drivers OLEDB para DB2, vai procurar e listar no combo da **Figura 11**. Ainda é necessário colocar o nome do schema antes do nome da tabela no select.

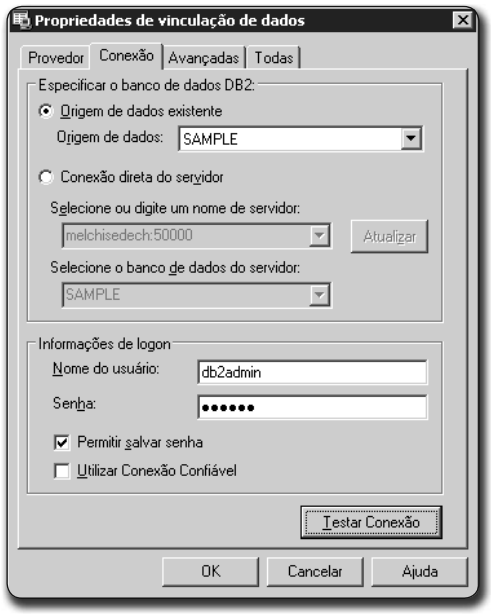


Figura 11. Conexão com ADO ao DB2 selecionando conexão existente

Colocando um provider próprio para db2, o TDB2UniProvider e fazendo-se as devidas alterações é possível configurar o UNIDAC para acessar o db2 apenas com um duplo clique no UniConnection. Uma vantagem é que, conforme visto na **Figura 12**, o UniConnection permite a escolha de um schema direto na conexão, mesmo estando logado com outro usuário, contanto que este tenha permissão de acessar o schema.

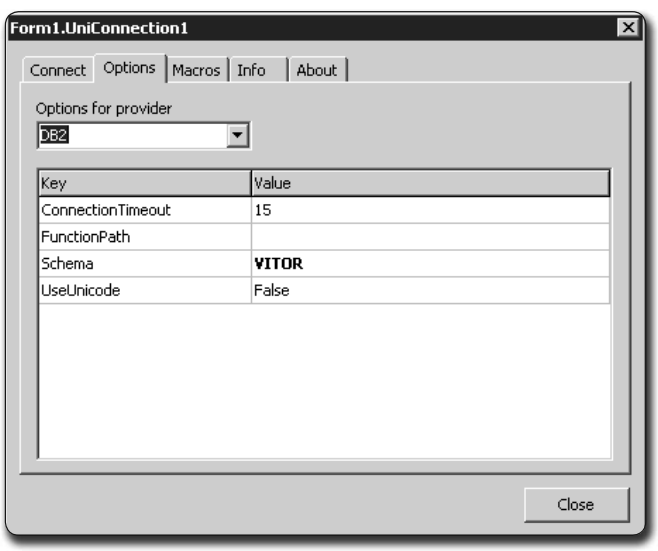


Figura 12. O UNIDAC permite a seleção do schema no objeto UniConnection

Com isso é possível acessar a tabela employee direto com a query `select * from EMPLOYEE`, sem se preocupar em informar o nome do schema. O prórpio UniConnection se encarrega disso.

Teste de execução

Apenas para exemplificar com um breve teste, a diferença de tempo de acesso aos dados com eles em cache ou acessando pela primeira vez é muito grande. Além disso há uma diferença grande também na velocidade de acesso com o DBX quando a propriedade GetMetaData do SQLDataset está false, que torna bem mais rápido do que quando está true, o padrão. Ao acessar pela primeira vez, temos o tempo de 5813 milisegundos para DBX (com GetMetaData = true), 438 milisegundos para ADO+OLEDB e 391 milisegundos para Unidac (**Figura 13**).

Já em uma segunda leitura, ou seja, com dados em cache, temos 0 milisegundos para o DBX e Unidac, mas 16 para ADO+OLDB. Desativando GetMetaData do DBX tem um resultado um pouco diferente. DBX obteve 3391 milisegundos, 453 para o ADO+OLEDB e 281 para o Unidac.

Nota-se nesses testes que o tempo de acesso com Unidac é o melhor de todos, quando não empata com as outras tecnologias.

Deve-se salientar porém que usando packed records -1 no client dataset fará com que ele obtenha todos os registros (fetch) logo que abrir o dataset, para depois fechá-lo e fechar as conexões. Depois disso os dados estarão em memória e trabalhar com eles fica muito mais rápido. Por isso que a abertura inicial torna-se mais lenta para o DBX tanto nos testes do artigo anterior como neste.

Usando-se packed records 10 e fetchondemand = true o ClientDataset trará apenas 10 registros, trazendo os próximos sob demanda à medida em que os dados vão sendo "navegados". Isso aumenta um pouco a velocidade, com a desvantagem de manter a conexão aberta. A escolha do método de acesso e cache mais uma vez depende de detalhes de negócio, projeto e implementação do seu sistema.

O código desses testes é idêntico ao da **Listagem 1**, pois são os mesmos botões, os mesmos componentes DBX, ADO e Unidac, mudando apenas o banco de dados acessado. Haveria diferença caso toda configuração de conexão, servidor, credenciais e drivers fosse feita em runtime.

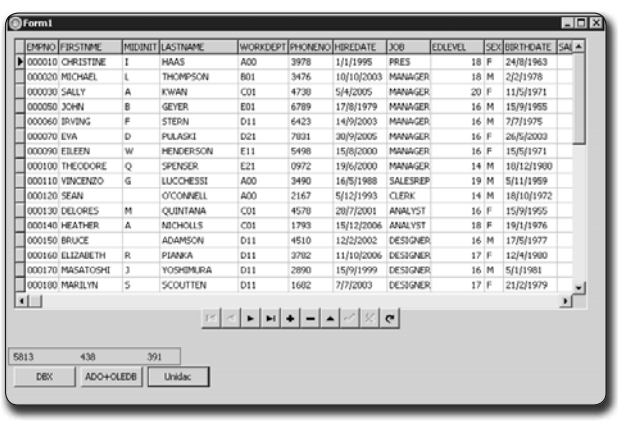


Figura 13. Tempo de acesso pela primeira vez, com GetMetaData = true

Nota do DevMan

Muito provavelmente ocorrerá um erro de BCD overflow ao se tentar conectar ao DB2, na tabela employee, com o DBExpress. Esse erro sempre acontece quando um campo do tipo float (Double, currency etc.) excede o tamanho máximo que pode ter. Na verdade os campos Salary, Bonus e COMM da tabela Employee não excedem esse tamanho máximo. O que acontece é um conflito de linguagem. No Brasil o separador de decimal é "," e de milhar é ".", enquanto que nos EUA é o contrário. O Delphi recebe do banco de dados um valor já no formato brasileiro, com vírgula separando os decimais, porém o DBExpress, no formato americano, encara a vírgula como separador de milhar, por isso entende que depois dela deve vir três zeros, multiplicando assim o valor que o banco de dados traz por mil. É o valor multiplicado por mil e não o valor original que "estoura" o máximo do BCD. Para corrigir esse problema deve-se editar a propriedade Decimal Separator dos parâmetros do SQL Connection, colocando vírgula como decimal separator no lugar de ponto.

Conectando-se ao Oracle Express

Embora a instalação do Oracle Express não tenha nenhum banco de dados de exemplo ele possui uma tabela HELP que tem a lista dos erros e motivos. Pode-se usar esta tabela para um teste de conexão.

Caso desejar você pode criar uma tabela de contatos por exemplo, para fazer o clássico sistema de agenda. Apenas consulte a documentação do Oracle para saber como criar campos com auto numeração, visto que, embora o procedimento para criar uma chave primária seja quase o mesmo em todos os bancos de dados, o procedimento para criar um campo auto-numeração não é padrão. Alguns bancos tem o tipo auto-numeração nativo, outros precisam que seja criado uma sequence ou um generator.

Para se conectar ao Oracle com o ADO existem dois possíveis providers que podem ser usados, o Oracle Provider for OLEDB, que estará lá caso tenham sido instaladas as bibliotecas client do Oracle, ou o Microsoft OleDB provider for Oracle, que é o driver da Microsoft.

Nota

Caso receba a mensagem de erro "ora-12154: tns: não foi possível resolver o identificador de conexão especificado" este problema pode ser causado devido ao fato de o oracle não ter acesso ao arquivo tsnames.ora. Nesse caso você deve ir à pasta C:\oraclexe\app\oracle\product\10.2.0\server\NETWORK\ADMIN e garantir que o usuário do Oracle acesse esse arquivo, ou liberar acesso para TODOS (everyone se for Windows em inglês). Outra situação muito comum desse erro é quando ele ocorre na tentativa de conexão em design time, no Delphi, com qualquer componente de acesso, mas não ocorre esse erro em runtime. Isso acontecerá caso você esteja usando o Windows 7 64 bits. O driver do Oracle não suporta nomes de arquivos com parênteses, e no Windows 64 bit o Rad Studio XE é justamente instalado na pasta "Arquivos de Programas (x86)". Para resolver esse problema pode-se reinstalar o Delphi em uma pasta sem parênteses, como "Arquivos de Programas" ou direto no "C:\Delphi". Outra solução paliativa é fazer uma cópia da pasta "Embarcadero", em "Arquivos de Programas (x86)", direto para "C:\ Embarcadero" e executar o bds.32 dali.

Pode-se usar as duas opções, aqui será ilustrado como usar o OLE DB Provider da Oracle. Para que esse provider possa ser usado é necessário instalar as bibliotecas client do Oracle, arquivo OracleXEClient.exe, que pode ser baixado no mesmo link que o banco de dados, mostrado na seção de links.

Entre os componentes encontram-se um OLE DB Provider, um ADO.Net Provider, que pode ser usado no C# ou Prism e um provider ODBC.

A diferença entre os dois providers é que o da Microsoft pede o nome do "servidor" (**Figura 14**) enquanto que o provider do Oracle pede o nome da "fonte de dados" (**Figura 15**). Como já mencionado, nos dois casos o nome é "XE" ou o nome que foi definido no tnsnames.ora.

Como usuário coloque system, a não ser que você tenha criado outro, e como senha, neste artigo, a senha definida na instalação para o usuário system foi 123456.

O CommandText do AdoDataset ficará "select * from help". Para criar uma outra tabela, talvez aproveitando parte dos arquivos SQL para criação de bancos de exemplo dos outros bancos vistos usa-se a "Homepage do banco de dados" disponível no menu Programas>Oracle Database 10g Express Edition.

Não é objetivo desse artigo explicar cada uma das opções do gerenciador do Oracle, mas clicando em SQL>comandos SQL>Informar comando, é possível criar mais tabelas para os testes.

Para se conectar ao Oracle com DBExpress são necessários os mesmos componentes de sempre: um SQLConnection, um SQLDataset, um Provider e um ClientDataset.

Além do login e senha as únicas opções a se configurar são o decimal separator, que deve ser vírgula, e o nome do banco de dados, que é XE (**Figura 16**). O command text do SQLDataset é o mesmo.

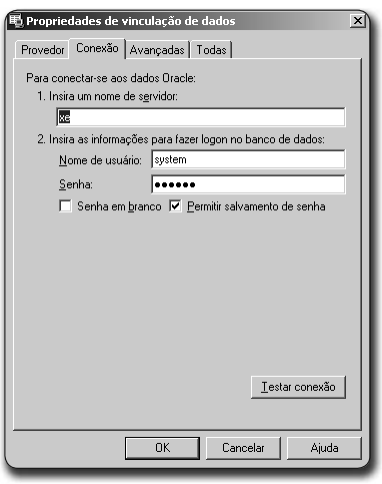


Figura 14. Provider da Microsoft

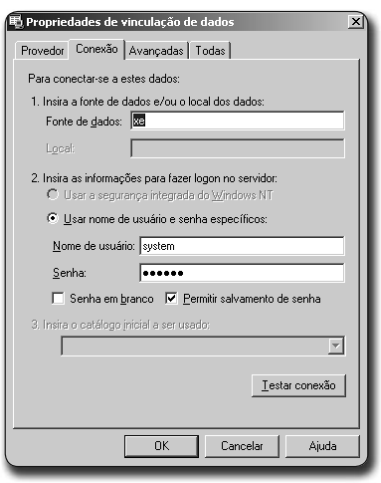


Figura 15. Provider da Oracle

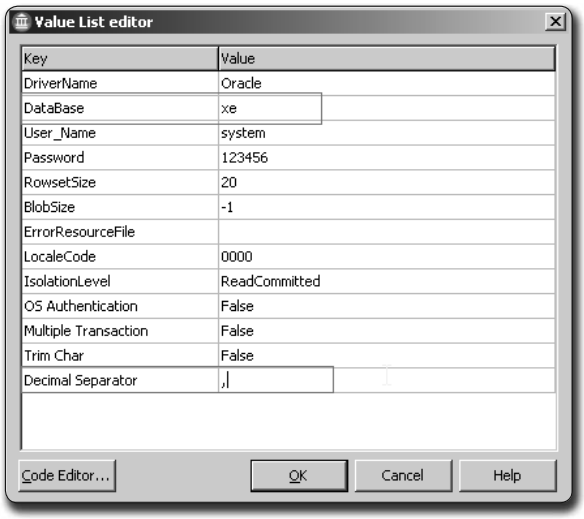


Figura 16. Configuração do DBExpress

A conexão usando o Unidac também é a mesma coisa. É necessário um UniConnection, um UniQuery, UniTransaction e um Provider do tipo OracleUniProvider. Com dois cliques no UniConnection aparece a janela de configuração que pede username, password e Server, que será XE.

Na aba options é possível acessar configurações avançadas, como Unicode e formato de data. Neste momento é possível perceber que os componentes de acesso ao Oracle pedem apenas o nome do banco de dados (ou instância).

Quem fornece esse nome de banco de dados é a biblioteca client, em conjunto com os arquivos de configuração ".ora" já mencionados. Esses arquivos que informam o ip ou host do servidor e a "instância" do banco de dados.

A **Listagem 2** mostra as conexões feitas com o Oracle usando DBX, ADO + Provedor Microsoft, ADO + Provedor nativo e Unidac. A única diferença em relação aos outros exemplos é o uso do ADO duas vezes, com providers diferentes.

Pode-se perceber com a execução desses três exemplos que o Unidac é uma grande (e atualizada) ferramenta, que se mostrou mais rápida na maioria dos casos, até mesmo em comparação com o DBExpress. É um bom investimento se for levado em consideração as vantagens. Seu custo é de aproximadamente 705 reais, a versão enterprise com source para um desenvolvedor. Há versões mais em conta sem source, e há versões para times de desenvolvedores.

Conclusão

Ao fim deste artigo pode-se concluir que o DBExpress pode acessar quase qualquer tipo de SGBD, mas não é a única opção, embora seja a mais indicada caso se queira portabilidade de banco de dados. Portabilidade de banco de dados não é uma coisa fácil de se fazer, e deve ser considerada desde a concepção do sistema, pois todas as consultas, tabelas e procedures devem ser feitas de maneira que funcionem em todos os bancos de dados ou refeitas em cada um deles.

Ao término dessa leitura espera-se que o desenvolvedor esteja apto a instalar e se conectar com os principais bancos de dados usados comercialmente em projetos Desktop ou web.

Listagem 2. Teste com Oracle

```
procedure TForm1.Button1Click(Sender: TObject);
var
 T: integer;
begin
 if ClientDataSet1.Active then
    ClientDataSet1.Close;
    t := GetTickCount;
    ClientDataSet1.Open;
    t := GetTickCount -t;
    Label1.Caption :=  IntToStr(t);
    DataSource1.DataSet := ClientDataSet1;
end;
procedure TForm1.Button2Click(Sender: TObject);
var
 T: integer;
begin
 if ADODataSet2.Active then
    ADODataSet2.Close;
    t := GetTickCount;
    ADODataSet2.Open;
    t := GetTickCount -t;
    Label4.Caption :=  IntToStr(t);
    DataSource1.DataSet := ADODataSet2;
end;
procedure TForm1.Button5Click(Sender: TObject);
var
 T: integer;
begin
 if ADODataSet1.Active then
    ADODataSet1.Close;
    t := GetTickCount;
    ADODataSet1.Open;
    t := GetTickCount -t;
    Label2.Caption :=  IntToStr(t);
    DataSource1.DataSet := ADODataSet1;
end;
procedure TForm1.Button6Click(Sender: TObject);
var
 T: integer;
begin
 if UniQuery1.Active then
    UniQuery1.Close;
    t := GetTickCount;
    UniQuery1.Open;
    t := GetTickCount -t;
    Label3.Caption :=  IntToStr(t);
    DataSource1.DataSet := UniQuery1;
end;
```

Tenha este artigo como um guia para cada vez que iniciar com um banco de dados novo, pois muitos desenvolvedores, embora tenham vasta experiência com um banco de dados, quando mudam de empresa, de projeto ou simplesmente de banco não sabem por onde começar.

Ao longo do desenvolvimento do sistema podem surgir, como sempre, problemas de tipos de dados, desafios de procedimentos armazenados, otimização de consultas lentas a tabelas muito grandes com indexação e uso de tabelas temporárias e todos os outros desafios que se enfrentam ao se desenvolver um sistema de informação com um sistema gerenciador de bancos de dados relacionais.

Recomendo consulta à literatura disponível para cada um desses bancos de dados além de acessar fóruns específicos de cada um para solução dessas dúvidas.

Não é intuito desse artigo formar administradores de bancos de dados, pois cada um dos seis bancos de dados vistos até agora pode ser esmiuçado em um ou mais livros. O foco principal desse artigo é apresentar formas diferentes de se acessar os mesmos bancos de dados, bem como fornecer uma espécie de "catálogo" para iniciantes em determinado SGDB, mostrando os passos iniciais para instalar e se conectar a cada um deles.

Vitor Luiz Rubio
vitorrubio@gmail.com - http://blog.vitorrubio.com.br
Tecnólogo em Processamento de Dados pela FATEC-SP e MCP. Atua como Analista de Sistemas Sênior na Editora Revista dos Tribunais. Programa com Delphi desde a versão 3 e com .Net desde a versão 1.1.

Microsoft SQL Server Express
http://www.microsoft.com/express/Database/

Windows Power Shell
http://www.microsoft.com/downloads/en/details.aspx?FamilyId=60cb5b6c-6532-45e0-ab0f-a94ae9ababf5&displaylang=en

Adventure Works Database
http://msftdbprodsamples.codeplex.com/releases/view/55926

Northwind e Pubs sample databases
http://archive.msdn.microsoft.com/northwind/

Oracle Database 10g Express Edition
http://www.oracle.com/technetwork/database/express-edition/overview/index.html

Oracle OLE DB Provider
http://www.oracle.com/technetwork/database/windows/utilsoft-088126.html

DB2 Express
http://www-01.ibm.com/software/data/db2/express/download.html

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Expressões Lambda

Com certeza você deve estar se perguntando, o que é expressão lambda. Que nome não é? Mas esse é o nome para um recurso muito útil que foi introduzido no .Net Framework em sua versão 3. De uma forma resumida posso dizer que uma expressão lambda é uma forma de se escrever métodos anônimos de forma mais clara e resumida.

Mais um conceito, talvez novo para alguns, precisa ser introduzido: métodos anônimos. Os métodos anônimos dispensam a criação de um método específico para operações que somente necessitam ser executadas uma única vez. Eles são utilizados em conjunto com Delegates. Como pode ser visto este é mais um conceito. Na verdade, expressões lambda são a evolução dos métodos anônimos que por sua vez são uma evolução dos delegates. Mas para entender como tudo isso se encaixa vamos explorar do início.

Delegates

De uma forma simplista dizemos que delegates são ponteiros para métodos dentro de uma classe. Cabe aqui uma consideração um pouco mais detalhada porque, na verdade, ponteiros não são uma coisa comum em .Net.

Delegates são objetos especiais que recebem uma referência para um método. São usados principalmente se precisarmos definir dinamicamente qual método será executado. Não existe necessidade de usar um delegate se o método a ser executado vai ser sempre o mesmo. Para criar delegates você precisa criar uma instância da classe delegate que define a assinatura do método a ser passado. Confira um exemplo da **Listagem 1**.

Primeiramente você declara um delegate com a palavra chave "delegate". Com isso, você define quais os parâmetros que o método deve ter. O método deve ser definido com os mesmos parâmetros que o delegate. Para poder usar o método, definimos uma variável do tipo do delegate que foi criado, e passamos para essa variável o métodos que será executado pelo delegate, no caso o ImplementacaoDelegate.

Métodos anônimos

Como já foi dito, os métodos anônimos dispensam a criação de um método específico para operações que somente necessitam ser executadas uma única vez. Tomando o exemplo anterior, podemos escrever o método diretamente ao instanciar o delegate criado. O código da **Listagem 2** mostra como ficaria o método anônimo para a mesma funcionalidade anterior.

Resumo DevMan

De que trata o artigo:

O .NET Framework está em constante evolução. Através do Delphi Prism, o desenvolver Delphi possui acesso total a novos recursos e melhorias que são adicionadas. Este artigo trata de uma das diversas novidades, as Expressões Lambda.

Em que situação o tema é útil:

No desenvolvimento diário é sempre melhor escrever um código de maneira mais simples, mais entendível e menos verboso. O uso das Expressões Lambda permite isso e muito mais.

Expressões Lambda:

Neste artigo vamos abordar a evolução de alguns recursos do .NET Framework. Tudo começa com os Delegates, passando então para os Métodos Anônimos e culminando nas Expressões Lambda, foco maior deste. Através de exemplos ficará claro como é importante evoluir uma linguagem, e como fazer uso dos novos recursos para criar um código mais legível.

Listagem 1. Como criar um delegate

```
namespace PrismLambda;
interface
type
  Program = class
  private
    class method ImplementacaoDelegate(mensagem: System.String);
    class method Main(args: array of System.String);
  end;

type
  MeuDelegate = delegate(mensagem: System.String);

implementation

class method Program.ImplementacaoDelegate(mensagem: System.String);
begin
  Console.WriteLine(mensagem);
end;

class method Program.Main(args: array of System.String);
  begin
    var d: MeuDelegate := @ImplementacaoDelegate;
    d('Teste de execução');
    Console.ReadKey();
  end;

end.
```

PAULO QUICOLI

Listagem 2. Métodos anônimos em ação

```
namespace PrismLambda2;

interface

type
 Program = class
 private
  class method Main(args: array of System.String);
 end;

type
 MeuDelegate = delegate(mensagem: System.String);
implementation
class method Program.Main(args: array of System.String);
begin
 var d: MeuDelegate := method (mensagem: System.String)
  begin
   Console.WriteLine(mensagem)
  end;
 d('delegate executado por método anônimo');
 Console.ReadKey();
 end;

end.
```

Listagem 3. Aplicando expressão lambda

```
namespace PrismLambda3;

interface

type
 Program = class
 private
  class method Main(args: array of System.String);
 end;

type
 MeuDelegate = delegate(mensagem: System.String);

implementation

class method Program.Main(args: array of System.String);
 begin
   var d: MeuDelegate := x -> Console.WriteLine(x);
   d('Delegate executado por expressao lamda');
   Console.ReadKey();
 end;

end.
```

Como pode ser visto não foi necessário criar um método específico com a assinatura do delegate, esse método foi criado no corpo do programa, sem ter uma declaração prévia. Por isso a esse tipo de método é dado o nome de anônimos, pois em uma declaração comum um lhe seria dado, nas como foi aplicado, ele não tem nome.

Expressões lambda

Foi falado que uma expressão lambda veio para simplificar a sintaxe de um método anônimo. A **Listagem 3** mostra como fica o código da **Listagem 2**, aplicando expressão lambda.

Se colocarmos lado a lado os exemplos vistos até aqui, fica nítida a simplicidade trazida pelas expressões lambda, embora sua sintaxe pareça um pouco "estranha" para alguns. Sua construção básica é feita com um sinal de traço seguido de um sinal de maior, como se fosse uma seta para a direita. Em inglês, lê-se esse sinal como "goes to", algo como "vai para". A construção `x -> x + 5` seria então lida como "x vai para x mais 5". Vamos explorar mais recursos e uso das expressões lambdas.

Lambdas como parâmetros

As expressões lambdas mostram seu real valor quando funcionam como parâmetros de funções. Nesse cenário elas se tornam verdadeiros pontos de extensão. O LINQ faz um uso extenso desse padrão e na verdade ele é um dos motivos que essa funcionalidade foi criada. Veja na **Listagem 4** um exemplo de uma função que espera um delegate como parâmetro. Foi criada uma lista (List<T>) com a chamada ao método de extensão ToList sobre um array de inteiros. A função em questão é a ForEach, da classe System.Collections.Generic.List<T> do .NET Framework, e ela possui a seguinte assinatura:

```
public void ForEach(Action<T> action)
```

Essa função recebe como parâmetro um delegate do tipo System.Action<T>. Este delegate possui por sua vez a seguinte assinatura:

```
public delegate void Action<T>(T obj);
```

O delegate Action tem esse nome porque foi criado com a intenção de simplesmente ocupar o lugar de um método que executa uma ação qualquer, e que recebe como parâmetro qualquer objeto. Qualquer função que receba um delegate Action como parâmetro, vai executar uma função qualquer que desconhece, qual seja, daí o ponto de extensão. Podemos injetar o código que quisermos nessa implementação. O código da função ForEach, por exemplo, tem como objetivo iterar sobre uma lista qualquer, e chamar a função passada no delegate Action em todos os itens da iteração,

Listagem 4. Expressão lambda como parâmetro

```
namespace PrismLambda4;

interface

uses
 System.Linq;

type
 Program = class
 private
  class method Main(args: array of System.String);
 end;

implementation

class method Program.Main(args: array of System.String);
 begin
    var numeros: array of System.Int32 := [1,2,3,4,5,6,7,8,9];
    numeros.ToList().ForEach(numero -> Console.WriteLine(numero.ToString()));
    numeros.ToList<System.Int32>().ForEach(numero -> Console.WriteLine
    (numero.ToString()));
    Console.ReadKey();
 end;

end.
```

Listagem 5. Expressão lambda com várias instruções

```
namespace PrismLambda5;

interface

uses
 System.Linq;

type
 Program = class
 private
  class method Main(args: array of System.String);
 end;

implementation

class method Program.Main(args: array of System.String);
begin
 var numeros2: array of System.Int32 := [1,2,3,4,5,6,7,8,9];
 numeros2.ToList().ForEach(numero -> begin
  if numero mod 3 = 0 then
   Console.WriteLine('É divisível por 3: {0}', numero.ToString())
  else
   Console.WriteLine('Não é divisível por 3: {0}', numero.ToString())
 end);
 numeros2.ToList().ForEach(method (numero: System.Int32) begin
  if numero mod 3 = 0 then
   Console.WriteLine('É divisível por 3: {0}', numero.ToString())
  else
   Console.WriteLine('Não é divisível por 3:{0}', numero.ToString())
 end);
 Console.ReadKey();
end;

end.
```

passando-os como parâmetro. Em uma função ForEach, o parâmetro é cada item da lista que recebe a chamada.

No primeiro ForEach podemos ver como o compilador foi capaz de inferir o tipo da variável "numero" como um inteiro, porque o delegate esperado era um Action<int> (ele foi chamado sobre uma lista de inteiros), então o parâmetro "numero" só poderia ser também um int. A inferência de tipo também aconteceu ao chamar o método ToList, que conseguiu "adivinhar" que devia criar uma lista de inteiros. No segundo ForEach temos o exato mesmo código, apenas sem utilizar a inferência de tipos, mostrando o grande valor dessa nova função da linguagem, já que permitiu simplificar bastante o código assim como diminuir a sua quantidade, deixando a expressão como um todo bem mais limpa.

Vimos o exemplo do poder de uma lambda: em uma única linha de código foi possível iterar entre os objetos de um array, e tomar uma ação sobre eles, que no caso foi escrevê-los na janela de comando. Esse tipo de Lambda, que executa uma única expressão, é chamada de Expression Lambda, ou Lambda de expressão.

Há também a possibilidade de criar Lambdas que se estendem por mais de uma instrução, chamadas Statement Lambdas, ou Lambdas de declaração. Veja na **Listagem 5** um outro exemplo que verifica se o número é divisível por 3, e escreve na console de acordo. Nesse exemplo, o delegate passado é uma função com várias instruções.

Conclusão

O avanço das linguagens não pode ser desconsiderado. É preciso estar ligado a cada nova versão para aprender sobre novos recursos e ver na prática como eles podem nos auxiliar. Com Delphi Prism o programador Delphi que quer utilizar o .NET tem agora em suas mãos os últimos recursos lançados para o .NET.

Paulo Quicoli
pauloquicoli@gmail.com
Editor técnico das revistas ClubeDelphi e .NET Magazine. Formado em processamento de dados pela FATEC-TQ. Atua como analista e desenvolvedor da Siplan Control-M (www.siplancontrolm.com.br) e consultor do NHibernate Brasil (www.nhibernatebrasil.net). Utiliza Delphi desde sua primeira versão, entusiasta do desenvolvimento orientado a objetos e UX, tem publicado vários artigos sobre o assunto, desde de boas práticas ao WPF, Silverlight e NHibernate. Blog: http://quicoli.wordpress.com

Introdução a Linq to Objects
http://www.codeproject.com/KB/linq/LINQtoObjects_CS.aspx

Lambda Expressions
http://msdn.microsoft.com/en-us/library/bb397687.aspx

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

DevMedia

DÊ UM SALTO EM CONHECIMENTO!

Acesse o maior portal para desenvolvedores da América Latina!

17 mil posts

280 mil cadastrados

5 milhões de page-views por mês

http://www.devmedia.com.br/

DevMedia

Hospede seus sites de forma simples, rápida e ilimitada.

Plano **Básico**

- 1 domínio principal
- 50 domínios hospedados
- Transferência **ilimitada**
- Espaço **ilimitado**
- 10 contas de e-mail
- PHP, ASP e ASP.NET
- Painel de controle Plesk
- Suporte 24hs

Tudo isso por apenas

R$ **15,90** por mês
pagamento trimestral

Assine agora mesmo e descubra por que Hospedix é a hospedagem de sites preferida entre os profissionais.

- Conheça também nossos planos profissionais com registro de domínio grátis e isenção de taxas anuais enquanto for cliente.

HOSPEDIX
HOSPEDAGEM PROFISSIONAL
www.hospedix.com.br

30 DIAS GRÁTIS

Ao assinar, digite o código de desconto exclusivo para leitores desta revista e ganhe 30 dias inteiramente grátis.

RVSTMEDIA

As promoções anunciadas não são cumulativas com outras promoções e são válidas apenas para clientes Hospedix ativos.